Naval Research Laboratory
Stennis Space Center, MS 39529-5004

NRL/MR/7180--02-8273

Comparison of 2D and 3D Predictions of the FOR3D Acoustic Propagation Model in Shallow Water Flat Bathymetry Environments

Allen E. Leybourne

Acoustic Simulation, Measurements, and Tactics Branch
Acoustics Division

and

University of Southern Mississippi

June 22, 2002

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*
June 22, 2002

2. REPORT TYPE

3. DATES COVERED *(From - To)*

4. TITLE AND SUBTITLE

Comparison of 2D and 3D Predictions of the FOR3D Acoustic Propagation Model in Shallow Water Flat Bathymetry Environments

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Allen E. Leybourne*

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Naval Research Laboratory
Acoustics Division, Code 7180
Stennis Space Center, MS 39529-5004

8. PERFORMING ORGANIZATION REPORT NUMBER

NRL/MR/7180--02-8273

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSOR / MONITOR'S ACRONYM(S)

11. SPONSOR / MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release; distribution unlimited.

13. SUPPLEMENTARY NOTES

*University of Southern Mississippi

14. ABSTRACT

In this study using the FOR3D Acoustic Model, little difference was found between the 2D and the 3D computational modes. The study was limited to shallow water flat bathymetry (along the slope) using sound velocity gradients normally associated with typical shelf/slope frontal environments as well as greatly inflated gradients. Several comparisons were made for specific cases (in 2D and 3D modes) having exact solutions, which by analogy provide clear demonstration of why substantial agreement was found for the two different modes.

Acknowledgment: This work was sponsored by the Office of Naval Research with technical management provided by the Naval Research Laboratory, Program Element 0603747N.

15. SUBJECT TERMS

FOR3D Model, shallow water acoustics, 2D versus 3D acoustic models, shelf slope fronts

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Allen Leybourne
Unclassified	Unclassified	Unclassified	UL	130	19b. TELEPHONE NUMBER *(include area code)* 228-688-4879

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39.18

TABLE OF CONTENTS

COMPARISON OF 2D AND 3D PREDICTIONS OF THE FOR3D ACOUSTIC PROPAGATION MODEL IN SHALLOW WATER FLAT BATHYMETRY ENVIRONMENTS

SUMMARY

Allen E. Leybourne

Investigations of The FOR3D Model of Ding Lee in the context of shallow water environments with flat bathymetry have shown that results from the Model's 2D computational mode are virtually identical to the 3D-mode. This result is somewhat surprising, in that, in the 2D solution mode, sound velocity (SV) gradients in the cross range horizontal direction are not utilized (assumed to be zero). Such close results require that horizontal refraction be adequately accounted for during computations in the 2D-mode. Arguments are presented through carefully crafted analogies that can be solved rigorously, demonstrating that the 2D-mode methodology actually accounts for in excess of 99% of the horizontally refractive effects.

The study, generally limited to shallow water, flat bathymetry has concentrated on sound velocity environments as predicted by the Princeton Ocean Model expected to have characteristics similar to those of shelf slope fronts. The dormant FOR3D Model code has been implemented on a DEC Alpha computer and supporting graphical codes for model output visualization generated.

To fully validate the conclusion above, many cases are presented having SV gradients greater than those anticipated for any realistic situation including one with gently sloping bathymetry. Beamforming on the model results also impressively demonstrated that the predicted horizontal refraction was identical for the two model modes.

The long computation time and extensive computer resources required for the 3D mode are not justified by the small improvement, if any, for ocean environments of interest to this study.

COMPARISON OF 2D AND 3D PREDICTIONS OF THE FOR3D ACOUSTIC PROPAGATION MODEL IN SHALLOW WATER FLAT BATHYMETRY ENVIRONMENTS

1. INTRODUCTION

For studies of acoustics effects in coastal environments, it is desirable to accurately model acoustic effects resulting from three-dimensional sound velocity (SV) fields. After environments of interest have been defined, the Princeton Ocean Model (POM) is capable of generating the associated SV fields. Study cases, suggestive of shelf slope fronts, were input to POM to generate the SV fields for study. Although it is anticipated that 2D and 3D acoustic models will interact differently with sloped bottoms, cases studied are for constant bathymetry since this study is for effects occurring in the water column only. Because of the shallow water conditions, 150 Hz is considered to be the minimum frequency that would have useful properties with even higher frequencies being desirable. As will be demonstrated later, computational demands imposed by higher frequencies impose serious constraints on what is practical.

Given that the 3D SV field and bathymetry are known, in order to study 3D SV field effects, an acoustic propagation model is required. Chosen for study was the FOR3D Model, version 1.1, 9.04.92, developed by Lee, Schultz, Saad, and Siegmann. The Model was available at the web-site:

ftp://oalib.njit.edu/pub/for3d/

It is reported to solve the parabolic form of the 3D wave equation and thus should be capable of modeling effects for environments with horizontal refraction arising from horizontal SV gradients.

As many detailed references, e. g., [1 - 6], are available outlining the derivation and description of the FOR3D Model, only a brief description will follow. Of the above, Reference 6 is the most definitive providing a pretty complete description of the computer model itself. The Model, a parabolic equation (PE) solution, enables calculation in either 2D-mode (zero cross range gradients) or 3D-mode (utilizes cross range gradients). As implemented, the solution is solved in cylindrical coordinates, marching the solution out in range in a cylindrical wedge. The calculations are along a prescribed set of radial vertical cuts. Step sizes on the order of 1/10 to 1/15 wavelength in range and depth are apparently critical to obtaining reasonable results. Although intuition dictates that an angular resolution equivalent to 1/10 to 1/15 wavelength through out the range should be used, particularly at maximum range, this requirement is less clear. None-the-less, this is the assumption made during this study.

Information available at the web site mentioned above, in addition to the source code, included sample input control data, SV fields, and bathymetry data. The example tested herein is referred to as the Harvard data set. No output stream

Manuscript approved October 18, 2001.

results from previous runs of the Model were available at the web site. However, it was possible to compile and run the downloaded code (after only minor modifications required by the local environment) on the DEC Alpha and the Cray90 computers. Identical results were obtained on both computers. The sample problem data sets were apparently intended only as examples for the user to get the Model up and running. The grid step sizes were much too coarse to provide any practical guidance as to the Model's efficacy. The limited experience with the Model above indicated severe demands on available computing resources would occur when applied to applications at 1000 Hz using the aforementioned grid step sizes. Therefore, this initial study is for 150 Hz cases only.

The Harvard SV fields are in Cartesian space, with other input data in a very restricted format. The user inputs myriad data, such as source location, bathymetry, bottom characteristics, etc. into the input stream. The input control data formats are very cryptic and difficult to relate to the documentation provided. Therefore, it was decided to provide the user with a much improved user interface. To accomplish this, while introducing the least coding errors, a set of preprocessing programs were developed to generate all input data streams in the Harvard input data format. Even though little of these efforts are of great technical interest, much of this detail is included in the Appendix for others who might have interest in using the Model.

2. OBJECTIVES OF THE STUDY

Examine whether or not calculations in 3D-mode produce results significantly different from the 2D-mode for flat bathymetry conditions. Where possible, independently validate these differences. If 2D-mode calculations are adequate, much resource time can be saved.

Improve the user interface and develop graphical outputs for visualization of the Model results (Programs for these purposes were not available from the web site mentioned above).

Couple the FOR3D Model with the SV fields from the Princeton Ocean Model (POM).

Define/develop the necessary computing resources required in support of this study.

3. OPERATIONAL CONSIDERATIONS AND REMEDIES

3.1 Selection of Computing Resource for 150 Hz Studies

At the Stennis Space Center there are several computer resources available each having their own operational characteristics. As the FOR3D Model is computationally intensive, the Cray computers at the site would appear to be the

systems of choice and were chosen initially. As initially coded, run times were extremely long, even for cases using coarse grids. Although the solution to that problem is of no particular technical interest, this first runtime bottleneck was traced to the fact that the entire SV and bathymetry fields were being read over and over. When this duplicity was removed, run times were reduced dramatically (by a factor of over 100).

As it was also possible to run the initial cases on the DEC Alpha comparative timings, after removing the aforementioned reading problem, showed that the DEC Alpha cpu time was about 1/5 of that of the Cray. Perhaps this should not have been surprising, as the FOR3D Model code is in Fortran 77. This code does not specifically address utilization of the Cray's parallel or vector processing capabilities. As this study is to determine applicability of the Model rather than its development, further study was conducted on the DEC Alpha, thus deferring any effort to reduce run time on the Cray.

3.2 Memory Requirements Consideration

Naval Research Laboratory personnel who have had considerable experience with numerical acoustic propagation Models indicated that for convergent model accuracy, step sizes for a numerical parabolic equation model should be at most 1/10 wavelength. This information was provided in the context of range and depth step size from experience with other 2D propagation models. Even the authors of the FOR3D Model in the descriptions that were examined were silent about the appropriate azimuth step size. One approach, the one taken herein, is to use an azimuth step size such that the radial separations at maximum range would be about 1/10 wavelength.

The effect of azimuth step size is illustrated by a few cases shown in Table 1. As the solution marches out in range, all of these grid points are not necessarily in memory at the same time, however, the 3D calculations must have the information for all of the radial grid points adjacent to the current range in memory at each range step. Thus, the problem scales directly with the number of radials.

On the DEC Alpha, it has been possible after purchasing a large hard drive that enabled increasing swap space, to run programs of compiled size approaching 2.3 Gbytes even though the RAM memory was only 512 Mbytes. An example case is one that used the specific conditions shown in Table 2. After achieving these results, run time on the DEC Alpha was deemed to be reasonable for the 150 Hz case. While the above results are fresh in mind, it was noted earlier that the 2D-mode calculations would save much time, particularly at 1000 Hz where the wavelength is about 1.5 meters. Admittedly 38 hr is not much less that 53 hr. and in of it self would not constitute a very significant benefit. However, in 2D-mode, the computational accuracy is independent of angular separation of the particular radials chosen. Therefore, angular spacing only as fine as needed for display of the results is required. For a case like this, instead of 2750 sectors,

only 1/10 to 1/20 as many would be needed, thus reducing run time to about 2 to 4 hr. a very manageable amount of time.

TABLE 1

Solution Grid Points Required For A 10 Degree Wedge; Depth, 195 M; Maximum Range, 15,000 M; Range And Depth Size, 0.1 Wavelength

Frequency, Hz	50	150	1000
Range Step, m	3	1	0.15
Depth Step, m	3	1	0.15
Cross Range Step, m*	3	1	0.15
Angular Step, degrees	0.0115	0.00382	0.000572
No, Range Steps	5000	15000	100000
No, Depth Steps	65	195	13000
No, Radials for 10 degrees	873	2618	17543
No, Solution Grid Points	284M	7660M	2270G

*At maximum range

TABLE 2

DEC Alpha Run Times for a Typical Set of Conditions

Range step size	1 meter, to a range of 15000 m
Depth step size	1 meter, to a depth of 195 m
Number of sectors	2750, covering a wedge of 10.0375°
Angular step size	10.0375°/2750=0.00365°
Cross range step size @ 15,000 m=0.955 m	
Resulting run time	3D-Mode= 52 hr. 2D-Mode=38 hr.

If it can be shown that the 2D-mode results are sufficiently close to the 3D-mode results, study of 1000 Hz cases is easily within the capabilities of the DEC Alpha system. From a computer science point of view, the FOR3D Model coded in Fortran 77, appears to be an unfortunate choice as the compiled size of such programs requires the entire program arrays to be allocated at compile time rather than dynamically when needed. This in turn places limitations on the program loader since it must allocate sufficient memory before the program is even able to load and start. Fortunately with sufficient swap space even very large programs can be loaded. The unfortunate side is that if the arrays are randomly accessed, processing speed would slow to a near halt condition. That this has not occurred with the FOR3D Model is a strong indication that the

interaction between the swap space controller and requests for array data by the program are well coordinated. Experience has shown that the extensive use of swap space has not been very deleterious to the operational time of the FOR3D Model once a minimally sufficient program kernel has been loaded into active memory.

In the event that the 1/10 wavelength step size would have to be maintained in all three dimensions for studies at 1000Hz, it is possible to conclude that the present DEC Alpha would be totally inadequate for that task. The 3D calculation case just presented requiring 53 hr. would quickly scale to at least 8000 hr. or more even if the program with the required larger arrays could be loaded, which is not possible with the present hardware. Fortunately, as will be shown later use of the 3D-mode will not be necessary, translating into a conclusion that the present DEC Alpha is adequate for flat bathymetry studies in the 2D-mode.

3.3 Choice of Boundary Conditions and Starter Function

As in all solutions to differential equations, the boundary conditions at the limits of the computational space must be provided in order for the solution to be obtained. The FOR3D Model anticipates this requirement, providing routines for automating that process. In addition to the usual requirement to choose bottom conditions, and surface conditions; conditions for the side-wall boundaries must be chosen. Little guidance is provided by the Model authors except to provide two built in functions to generate the side-wall conditions: (1) pressure release and (2) results along the side-wall boundary from the 2D-mode solution of the Model. Clearly condition 1 is not the condition in any real ocean. Preliminary trials using this option produced results that are truly difficult to accept, except for the heart-cut radials, i. e., the results far removed from the side-wall, near the wedge center radials. Condition 2 seemed to be a much more reasonable choice, although this condition would be expected to bias results toward that of solutions obtained using the 2D-mode. However, in that case, one might expect the heart-cut values to be closer to reality.

The Model also provides an option wherein the user can supply a different set of side-wall conditions. This seems to beg the question, after all, the reason one uses the Model at all is to answer the question, "What is the resulting three-dimensional acoustic field?"

For these studies, a Green's starter function and condition 2 above, i. e., side-wall conditions predicted by the 2D-mode, were used.

4. THE PRINCETON OCEAN MODEL AND OTHER DATA SETS

The original FOR3D Model (using the Harvard input data format) apparently was intended to access its ocean database using latitude and longitude for the data domain and source location. In addition, the propagation direction as defined by the central radial of the computational wedge is in reference to geographical

north. Though the domain is rectangular, in general, its Y-axis may have alignments other than North.

The POM Model used in this study produces a sound velocity field whose plan view is rectangular, without reference to latitude, longitude or azimuth of the domain. For purposes of compatibility, the preprocessing program harv_build.f requires as input the domain center (latitude/longitude) and domain extent. The domain is forced to have its Y-axis aligned with north, therefore, as the sound velocity fields shown herein are viewed, North is the positive direction of the Y axis. The propagation direction of the central radial is the offset direction from the Y-axis with clock-wise offsets positive (+).

For persons who do not need to understand these details, the calculation wedge of the FOR3D Model for different cases is shown as a fan shaped overlay on the top layer of the appropriate sound velocity fields, e.g., see Fig. 8. The sound source is located at the apex of the wedge.

4.1 The POM Cases

The POM study cases were selected to be analogous to those thought to be present in shelf slope fronts. The SV fields were computed for an 8-day simulation sequence. Of these, days 0 and 8 were selected for study. The colorbar range shown for the POM Model data plots includes the entire sound velocity range, i. e., all of the data are visible in those plots. Several additional sound fields were generated with the intention of exaggerating the sound field gradients in order to obtain greater horizontal refraction effects.

4.1.1 Day 8

This field of day 8, shown in Fig. 1, has a feature of the greatest interest to the coastal acoustics research. Note that a fairly well defined, pinched off vortex, is visible, particularly in the upper layers. Fig. 2 is a plan view of the top layer only wherein the field variations are the most pronounced. The colorbar scale at the right of the figure covers the entire range of the data in this sound field.

4.1.2 Day 0

The front present in the sound field shown in Fig. 3, contains the greatest organized horizontal SV gradient for any of the 8-day simulation sequence. Figure 4 is a plan view of the top layer similar to the presentation in Fig. 2. Here, the colorbar scale also covers the entire range of SV data. In this top layer, the maximum gradient is about (1504-1490)/10000=0.0014 sec^{-1}

4.2 Artificially Generated SV Fields

4.2.1 Vertical Copy of the Surface Layer of the Field Shown in Fig. 4

By copying the top layer of the SVs shown in Fig. 4 vertically through out the water column, the field shown in Fig. 5 was generated. It represents a case having a much stronger overall possibility of producing different 2D and 3D-mode results since the stronger gradient of the top layer exists through out the water column.

4.2.2 SV Field of Constant Cross Range Horizontal Gradient, -0.01 sec^{-1}

The entire SV range of this arbitrarily generated SV field of Y direction gradient of -0.01 sec^{-1} is shown in Fig. 6. Only the top layer is presented as all of the other layers are identical. The gradient is considerably greater than any gradient present in the SV field shown in Fig. 4. For easier comparison, Fig. 7 is a re-plot of Fig. 6 where the new color scale is the same as that of Fig. 4. Though field gradients of this strength do not exist over wide regions in any of the study areas of present interest, it is clear that any horizontal refractive effects impinging on the results of the FOR3D Model will be greatest for this data set.

4.2.3 Day 8, SV Field, Compressed Horizontally by a Factor of 4

The SV field of Day 8 was compressed horizontally by a factor of 4. The vertical gradients remain constant, whereas the compression has the effect of increasing the horizontal gradients by the compression factor. Close examination of Fig. 2, the original field, shows that there is a region of SV minimum through the vortex in the direction of about 10:00 o'clock.

4.2.4 Day 8, SV Field, Compressed Horizontally but Having an Inverted Gradient

The SV field of Day 8 was compressed horizontally by a factor of 4 and the gradient of each layer inverted as follows:

1. The average value is found for each layer.

2. The difference from the average for each point of the layer is found,

3. The average is added to the inverted differences of the layer, forming the new layer.

Figure 1

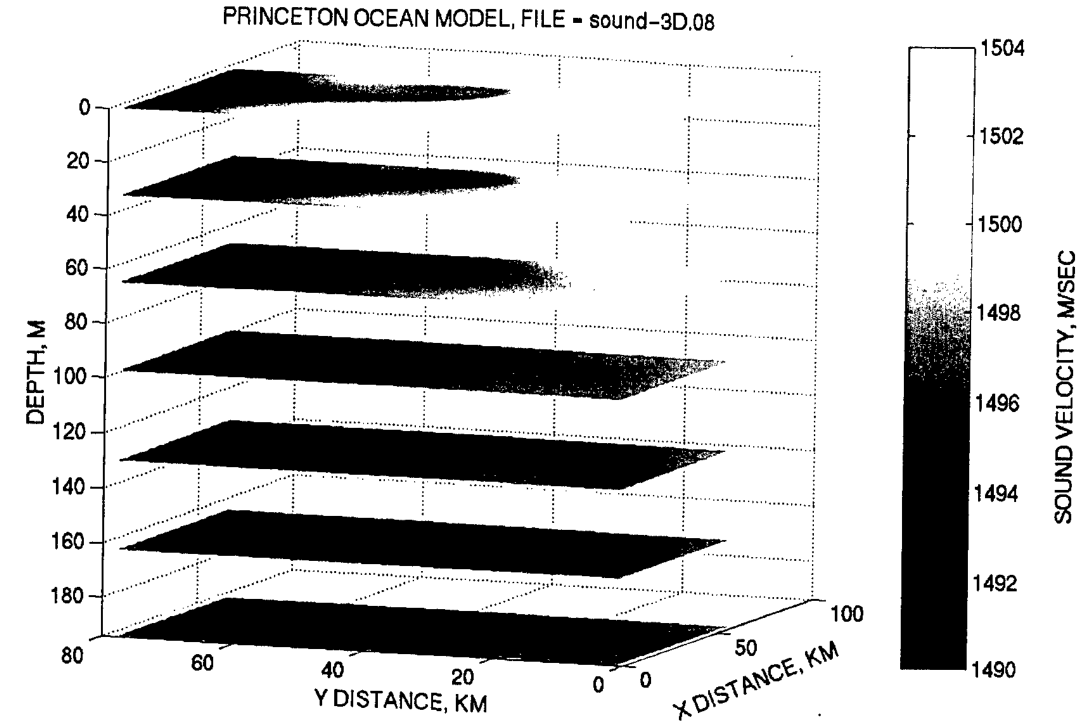


Figure 2

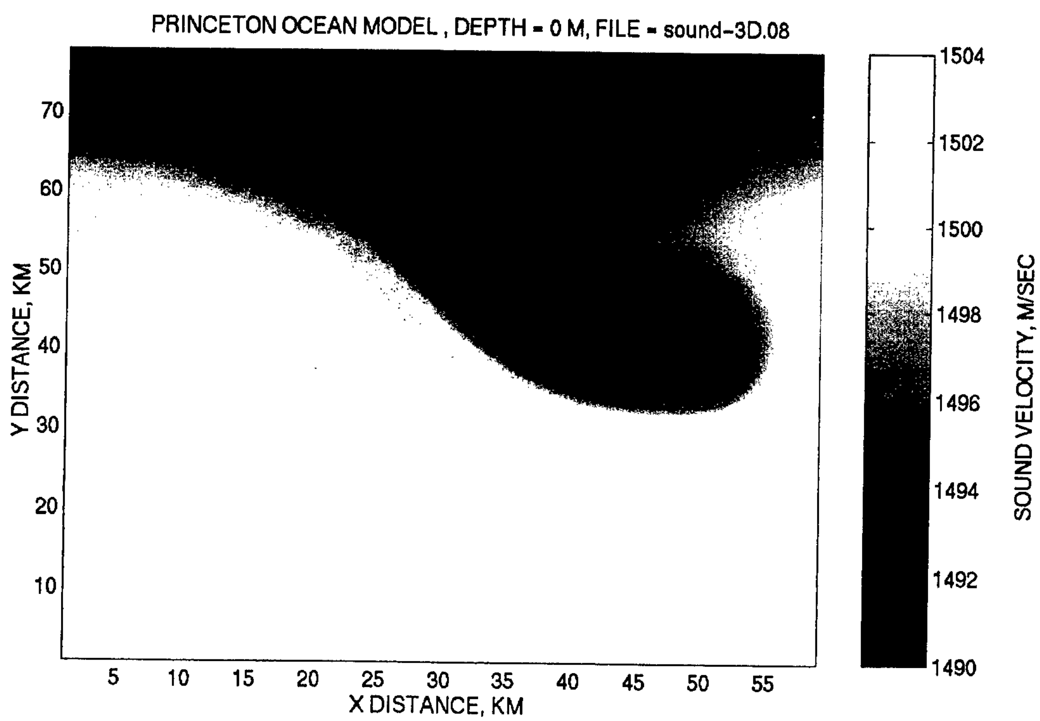


Figure 3

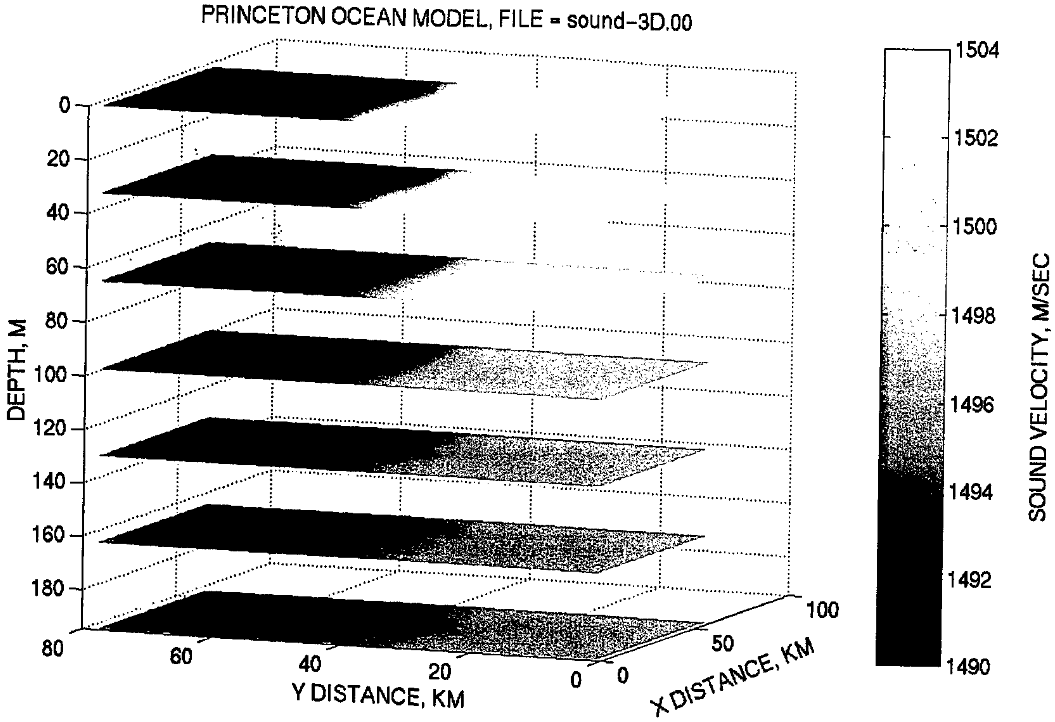


Figure 4

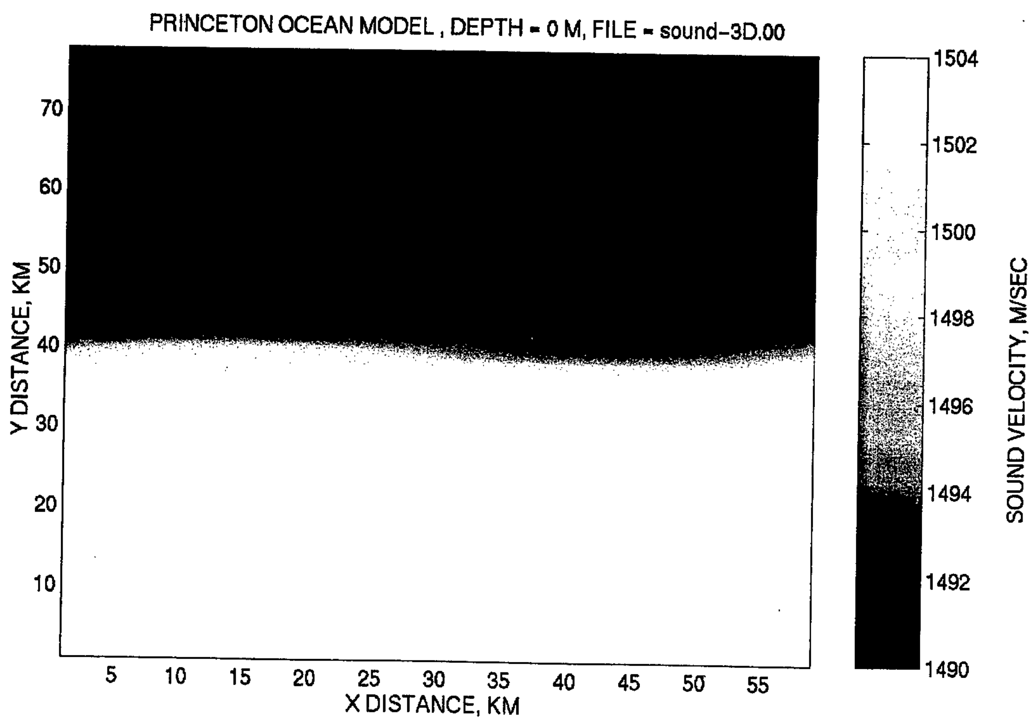


Figure 5

PRINCETON OCEAN MODEL, FILE = sound-3D.00T

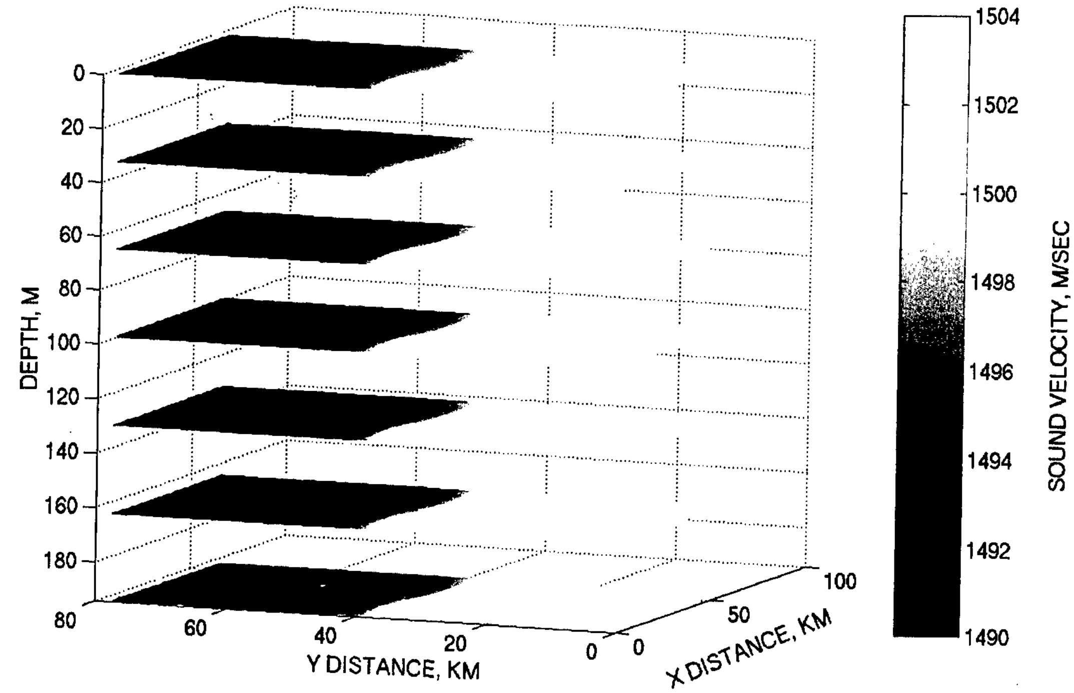


Figure 6

PRINCETON OCEAN MODEL , DEPTH = 0 M, FILE = sound-3D.grad10

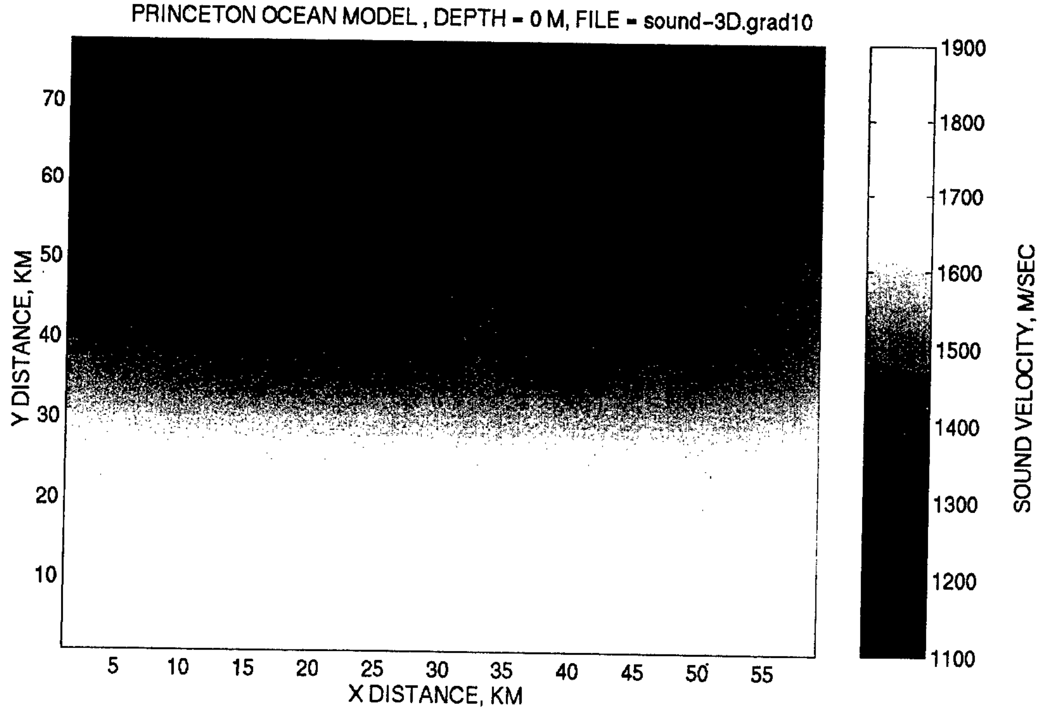


Figure 7

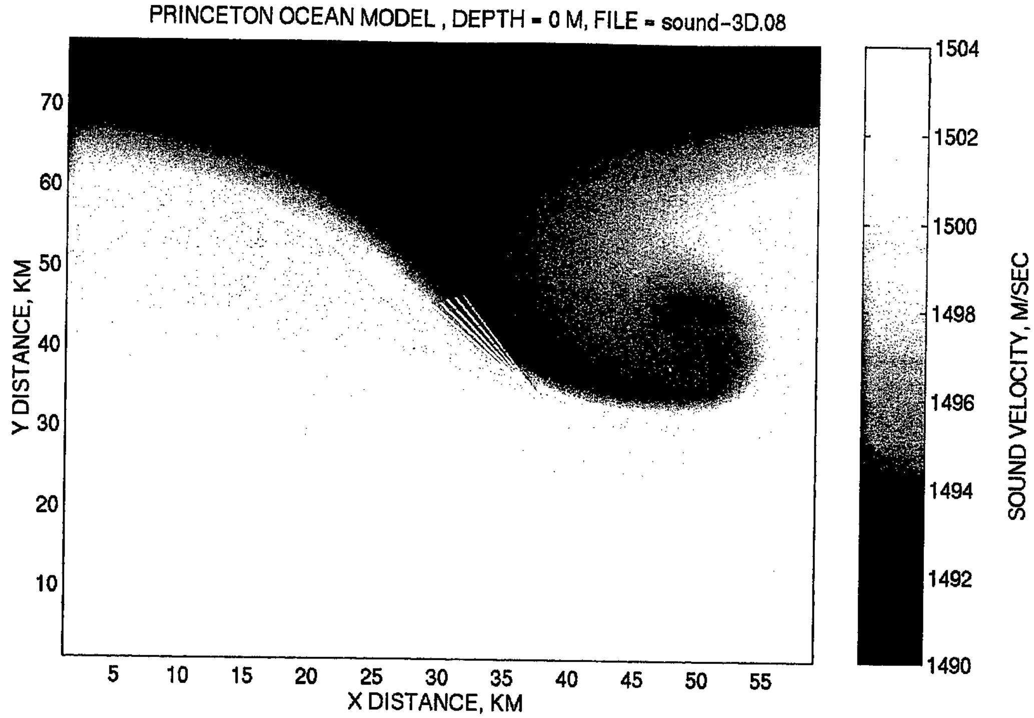
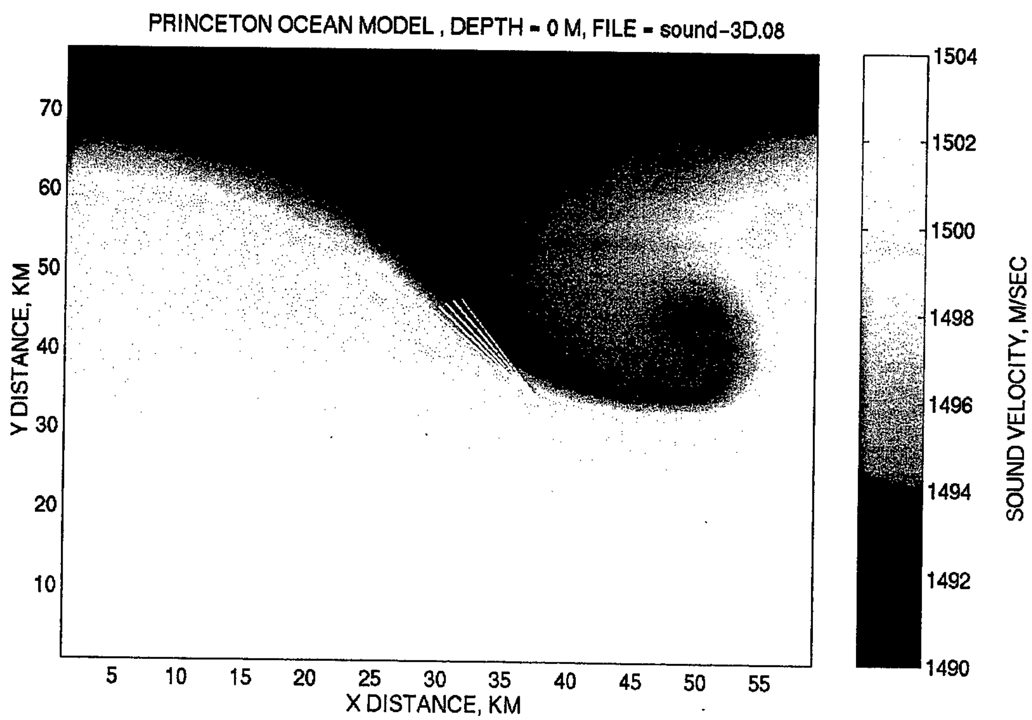


Figure 8



The vertical gradients, though inverted, have the same absolute amplitude as before. The compression has the effect of increasing the inverted horizontal gradients by the compression factor. Close examination of Fig. 2, the original field, shows that such an inverted field will have a region of SV maximum through the vortex in the direction of about 10:00 o'clock in the same region as the above field.

5 DETAILED COMPARISON OF RESULTS FROM FOR3D MODEL 2D AND 3D-MODES USING FLAT BATHYMETRY

In this section, the results predicted from 2D and 3D-modes are compared using common input parameters. Exceptions are the specific sound velocity fields chosen, the source locations, the propagation direction and, of course, the two different modes.

5.1 Standardized Model Run Time Parameters

The standardized conditions are shown in Table 3, which are virtually identical with the input stream file harvard.cfg. The interaction between these parameters is reasonably complex, in fact, some combinations of switches are incompatible and/or the program may crash for reasons not fully understood.

Some of the values in Table 3 may be over ridden when the preprocessing program, harv_build.f is run. Non-the-less, most of the key parameters can be correctly determined from this table.

5.2 Fields with Approximately Linear Cross Range SV Gradients

5.2.1 <u>Results for Propagation through the POM Day 8 SV Field</u>.

The POM Day 8 Sound Field, suggestive of shelf slope fronts of main interest to this study, is shown is Figs. 1, 2, and 8. The computational wedge shown in Fig. 8 has its central axis aligned perpendicular to the strongest gradient present in the field. This location appears to provide an opportunity to determine if calculation in 3D-mode produces results different from that obtained with 2D-mode.

Figures 9 and 10 present the dB transmission loss (dB TL) predicted for the 2D and 3D-modes respectively. Propagation is along the central radial of the computational wedge shown on Fig. 8 with source at the wedge apex. Even though these two color plots are virtually indistinguishable, close examination does reveal minor differences.

A horizontal slice at the 100m depth (the sound source depth), provides a more sensitive comparison. Figure 11, just such a plot, compares the 2D and 3D-modes.

Table 3 (Sheet 1 of 4)

FOR3D Model Input Parameters, Annotated as They Appear in Input Stream File, "harvard.cfg"

3	NDIM 1 - Model generates 2D solution 2 - Model generates NSOL x 2D solutions 3 - Model generates NSOL, 3D solutions
150	FRQ, Frequency, Hz
100.00	ZS, Source depth - meters
1500.0	C0, Reference speed of sound - meters/sec If C0=0, C0 is set to avg. of first layer
2	ISF, Starting field flag 0 - Program generates a gaussian starting field at range = 0.0. See SUBROUTINE SFLD3D. 1 - User supplies starting field. See SUBR. USFLD3D 2 - Greens wide angle starter 3 - SPARE
0.00	RA, Horizontal range, from source to starting field - meters. RA is set to 0.0 if starting field is Gaussian. RA is incremented by DR as solution is marched out in range.
195.00	ZA, Depth of field at range RA - meters If ZA=0, set ZA to max depth of bot. layer 1st profile Else, initial depth of starting field at range RA is: If ITYPEB = 0 or 1, set ZA to maximum depth of bottom most sediment layer at initial range of starting field. If ITYPEB = 3, ZA is maximum depth of artificial absorbing layer at initial range of starting field. Program inserts layer. RHO and BETA are obtained from layer above. Speed is bottom-most speed from layer above. Bottom of absorbing layer remains flat.

Table 3 (Continued, Sheet 2 of 4)

195	N, Number of equi-spaced receivers in U U is array - complex acoustic pressure field. Includes bottom point - but not surface point. If N=0, N is computed at 1/10 of wavelength.
0	IHNK, Hankel function flag 0 - Hankel fun. not used. 10*log(R) idded to solution. 1 - Starting field divided by Hankel function. Solution multiplied by Hankel function before computing propagation loss.
0	ITYPES, Type of surface 0 - Press. release. SCON3D sets SURY and SURX = 0.0 1 - User supplies surface condition, SUBR.USCON3D 2 - SPARE
3	ITYPEB, Type of bottom 0 - Pressure release, BCON3D sets BOTY and BOTX = 0.0 1 - User bottom condition. See SUBR. UBCON3D. 2 - SPARE. 3 - Absorbing layer used - bottom of layer is flat 4 - SPARE.
2	ITYPPW, Type port side-wall boundary cond. 0 - Field along port side-wall is set to 0.0. 1 - User supplied. See SUBROUTINE UPORT3D. 2 - Model generates 2D Solution if NDIM = 3.
2	ITYPSW, Type stbd side-wall boundary cond. 0 - Field along stbd side-wall is set to 0.0. 1 - User supplied. See SUBROUTINE USTBD3D. 2 - Model generates 2D Solution if NDIM = 3.
10.0375	FLDW, Width of field, degrees. Ignored if NDIM=1
2750	NSEC, Number of sectors in field. Ignored if NDIM=1 Number of solutions, NSOL=NSEC+1
15000.00	RMAX, Maximum range of solution - meters.

Table 3 (Continued, Sheet 3 of 4)

1.0	DR, Range step - meters If DR = 0, DR is set to 1 meter, then if bottom is not flat, DR is recomputed so that max depth is incremented or decremented by DZ
250.00	WDR, Range step solution is output - meters WDR is rounded to nearest DR
1.00	WZ1, First receiver depth solution is output
194.00	WZ2, Last receiver depth solution is output In other words, write WZ1 to WZ2 by WDZ - Meters
1.0	WDZ, Depth step solution is output - meters WDZ Selected so that plot program does not interpolate between widely spaced receivers. WDZ rounded to nearest DZ
0.01825	WDTH, Azimuthal step soln. is output - deg WDTH rounded to nearest DTH
100.00	PDR, Range incr. for soln. output - meters PDR rounded to nearest DR
1.0	PDZ, Depth incr. for soln. output - meters PDZ rounded to nearest DZ
0.5	PDTH, Azimuthal incr. for soln. output - degrees. PDTH rounded to nearest DTH
1	ISFLD, Starting field print flag 0 - Do not print starting field 1 - Print starting field
0	ISVP, SVP print flag 0 - Do not print sound velocity profile 1 - Print sound velocity profile
0	IBOT, Bottom depth print flag 0 - Do not print bottom depths 1 - Print bottom depths

Table 3 (Continued, Sheet 4 of 4)

0.00	DOUGRA, range flag 0 - Use Crank-Nicolson method a #, Is the range to switch to the Douglas method – meters
5	NDIV, If Douglas method requested, divide N by NDIV., NDIV=5 is recommended.
1.000	Set U1=1 for ocean Model data set
65.3430	SLAT0=U2, Latitude of starting field
41.312	SLNG0=U3, Longitude of starting field
-30.000	DIR=U4, Direction of propagation of center ray
1.500	BOTRHO=U5, Density in bottom
0.000	BOTRHOG=U6, Density gradient in bottom
0.500	BOTBETA=U7, Attenuation in bottom
0.000	BOTBETAG=U8, Attenuation gradient in bottom
0.965	CWCB=U9, SS ratio at bottom interface, CW/CB
1.700	CGRAD=U10, Sound speed gradient in bottom
200.000	SEDZ=U11, Sediment thickness
0.000	U12 = Spare, Currently unused

Examination shows that even though the results as a function of range are not identical, the differences are insignificant. Note, that calculations for the model are for a fixed (time invariant) sound field. The agreement between the two different mode predictions is closer than would be expected between successive sound fields resulting from only minor temporal changes in the environment.

Upon closer examination of Figs. 9 and 10, one can note that, viewed vertically, at specific ranges, the color transitions of the dB TL rather smooth variations on the 2D plots are replaced by uneven (jittery) transitions on the 3D plots. These color plots are difficult to compare quantitatively. Comparison is somewhat easier in Fig. 12, where vertical slices at 12000 m range are overlaid for the two computational modes. The magnitudes are seen to be virtually identical. However, there are 3D-mode oscillations about the smoothly averaged 3D curve. These minor oscillations are probably computational artifacts from the more complex algorithms that are used by the 3D-mode rather than real effects deriving from fundamental acoustic interactions. The FOR3D Model authors hint at computational instabilities than can occur, however, the remarks are somewhat general and shed little light on the subject under discussion here.

Lee (1994) [7] reports significant differences between 2D and 3D results for this Model while propagating through a pinched off eddy of the Gulf Stream (SV data via simulation using the Harvard Open Ocean Model) with propagation orientation similar to the case being presented here. In that study source frequency was 50 Hz with radial separations of 1° and propagation up to 100 km. In this discussion, Lee concludes that the 3D-mode captures effects that are not determined from the 2D-mode, citing differences as large as 8 dB at ranges greater than 40 km. The propagation track was in a region that had a strong cross range bathymetry gradient, generally down-sloping. Such regions would be expected to produce different results for 2D and 3D-modes simply because of the bottom bathymetry. Although the data as presented are difficult to read, it appears that the cross range SV gradient at 15 km range is about 0.0005 to 0.001 sec^{-1}, which is similar to that of this study.

5.2.2 Results for Propagation through the POM Day 0 SV Field.

The POM Day 0 Sound Field, Figs. 3, 4 and 13, contains the greatest organized horizontal SV gradient for any fields of the 8-day simulation sequence. The computational wedge chosen has its central axis aligned perpendicular to the strongest gradient present in the field, i.e., parallel to the front, with its apex location at the extreme left side as shown by the wedge in Fig. 13. This location provides a somewhat larger gradient than that of the location used for the Day 8 sound field.

Figure 9

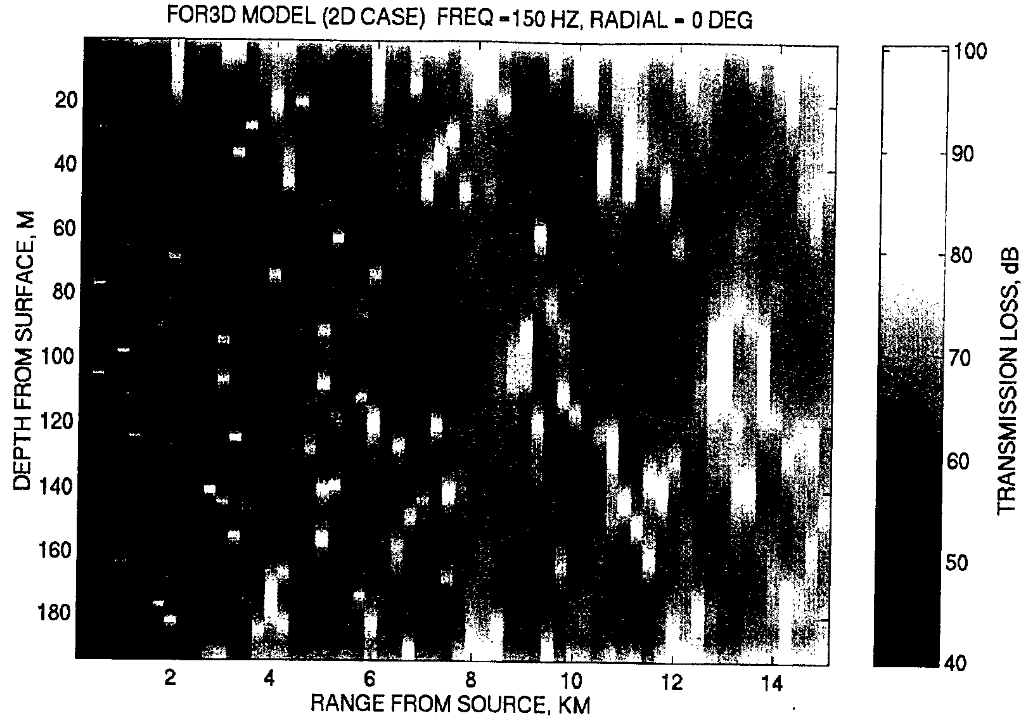


Figure 10

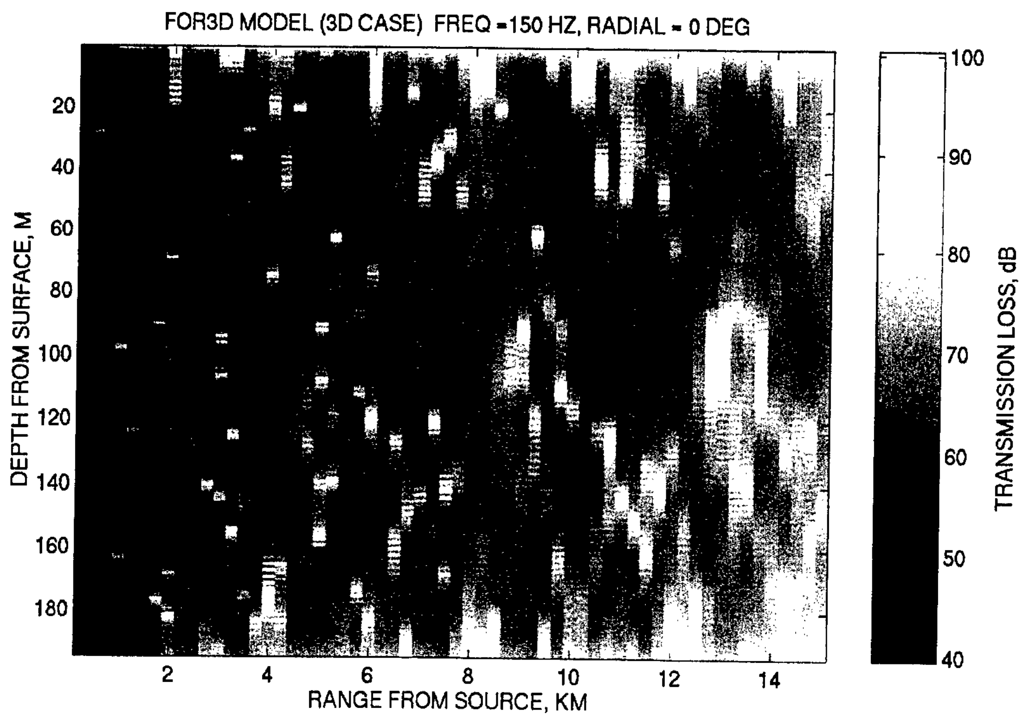


Figure11

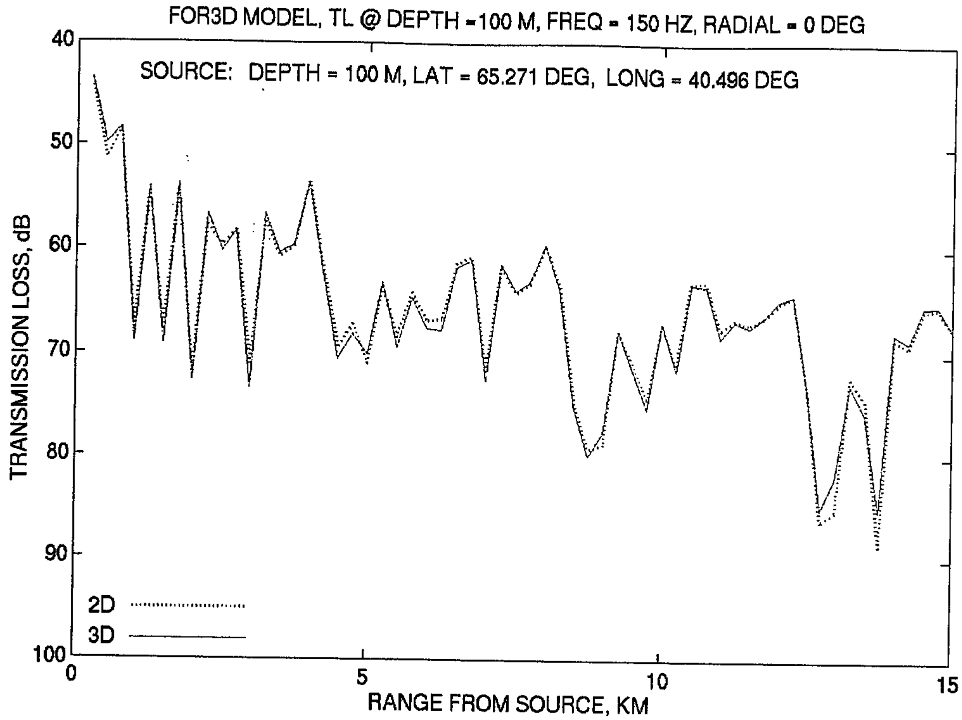


Figure 12

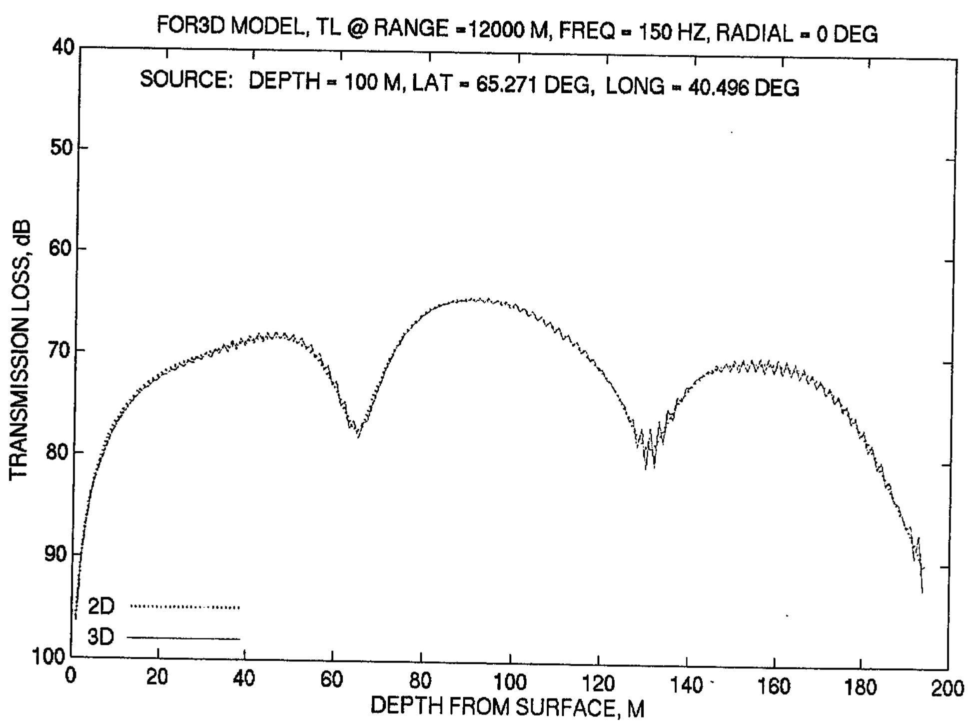
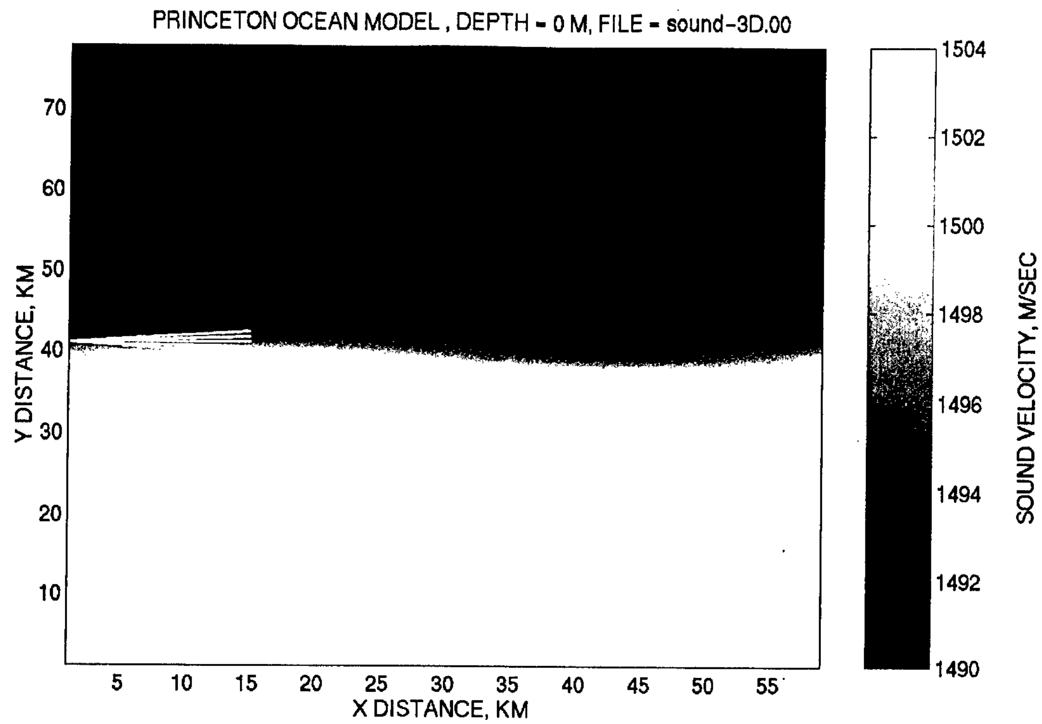


Figure 13



Results of the FOR3D Model were obtained as before. The differences between the dB TL color plots are similar to those for Day 8 and are therefore omitted. Only minor differences are seen in Fig. 14 presenting the dB TL predicted for the 2D and 3D-modes for a horizontal slice at the 100m depth. In Fig. 15 the dB TLs at 12000 m range are compared. Here, the minor 3D-mode oscillation is observed. Though the results are not identical, the differences are again insignificant.

Examination of Fig. 3, the 3D SV field, shows that the gradients present at the surface dissipate rapidly with depth. This raises the question, "Is the lack of 3D (horizontal refraction) effect caused by this attenuation of the SV gradient with depth?"

5.2.3 <u>Results for Propagation through an Artificially Generated POM Day 0 SV Field.</u>

By copying the SV values from the top layer of the POM Day 0 field to all depths field, a field having even greater cross range refractive effects should be produced. This field is shown in Fig. 5. Source location and propagation paths are again located as shown in Fig. 13.

Results of the FOR3D Model were obtained as before. The differences between the dB TL color plots are similar to those for Day 8 and are therefore omitted.

The minor differences can be seen in Fig. 16, the horizontal slice, and Fig. 17, the vertical slice. Again, the differences are insignificant. The 3D-mode oscillations are somewhat greater for this case. This slight increase is probably caused by the process of copying the larger gradients of the top layer vertically throughout the water column.

This result suggests that an even more extreme test should be made.

5.2.4 Results for Propagation through an Artificial SV Field with Gradient of 0.01^{-1}sec.

Figures 6 and 7 show an artificially generated SV field having a cross-range gradient of -0.01sec^{-1}, i.e., a gradient approximately 10 times that of any expected in the environments of interest. The color scale of Fig. 7 is identical to that of the POM SV field plots. The increase in gradient is obvious. Figure 6 is plotted with a different color scale that includes the entire data range, shows that the gradient is constant for the entire span of Y.

Though the field is different, orientation of the computational wedge was the same as shown in Fig. 13, i.e., from left to right. Because of the strong gradient imposed here, it was expected that differences between the 2D and 3D-modes would be fairly obvious. The differences between the dB TL color plots are similar to those for Day 8 and are therefore omitted. The minor differences that do exist can be seen in Fig. 18, a horizontal slice, and Fig. 19, a vertical slice. Again results were nearly identical.

5.3 Discussion of Cases having Linear Cross Range Gradients

5.3.1 Differences between 2D and 3D-Modes

In addition to results presented in this report, many trials were made where the radial separation was much greater than 1/10th wavelength at maximum range. Essentially no difference was noted between the 2D and 3D-modes. Since the differences found were minimal at 1/10th wavelength, it is fairly obvious that errors produced from too large an azimuthal step size would go unnoticed.

In all of the cases examined thus far, the horizontal cross range gradients were nearly constant. Especially note that there was no maximum or minimum in the cross range SVs. At this point one becomes reasonably convinced that, if any differences between the two modes exist, they are not discernable for the parameters used in situations exhibiting relative constant cross range gradients.

Figure 14

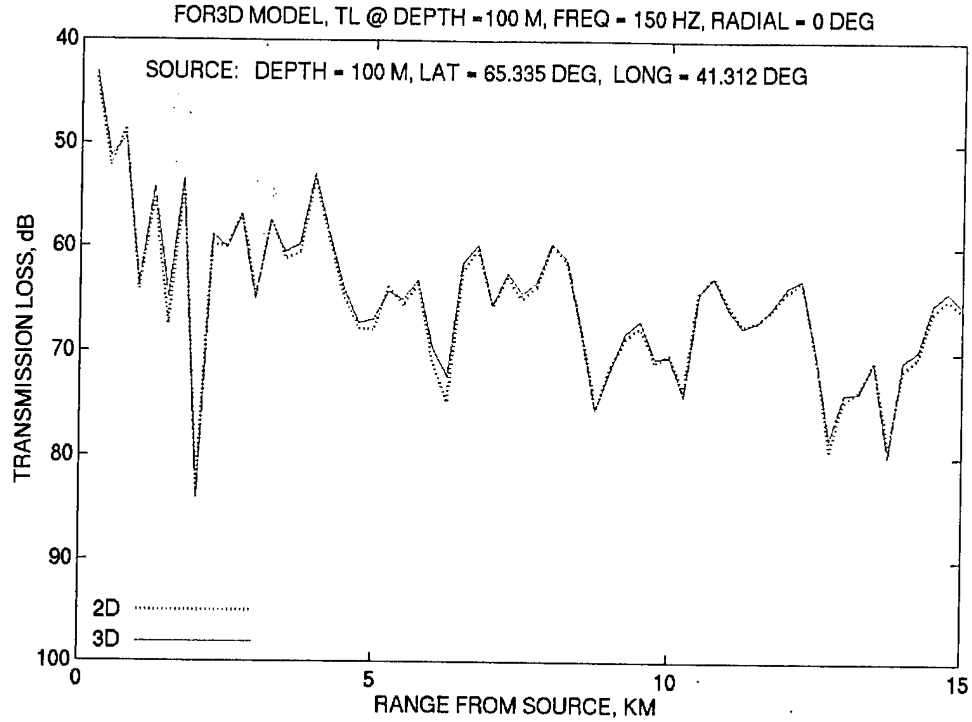


Figure 15

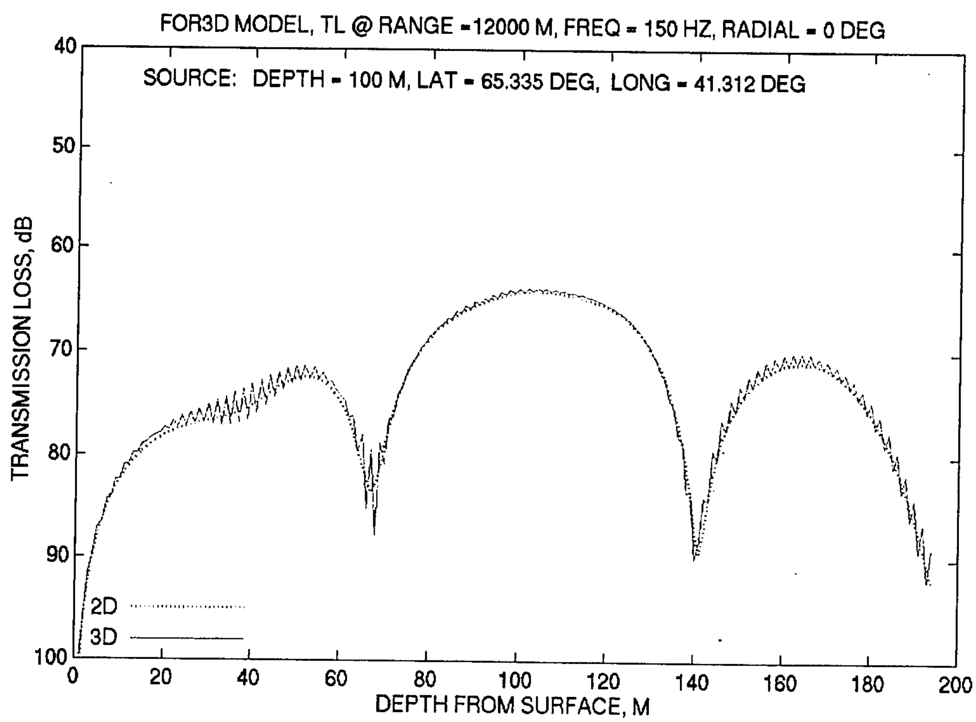


Figure 16

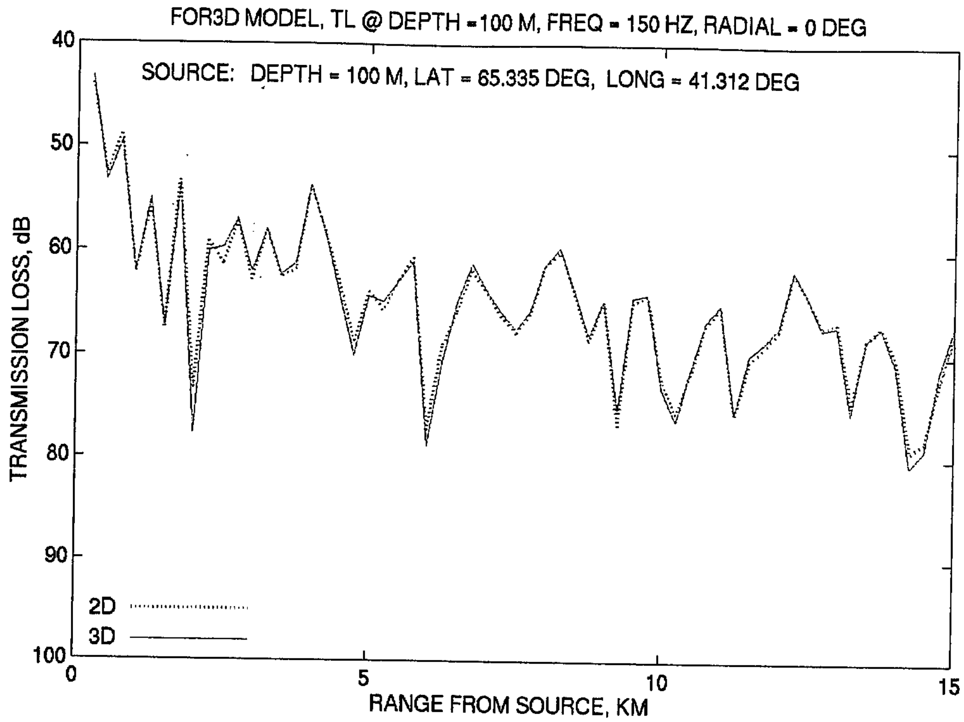


Figure 17

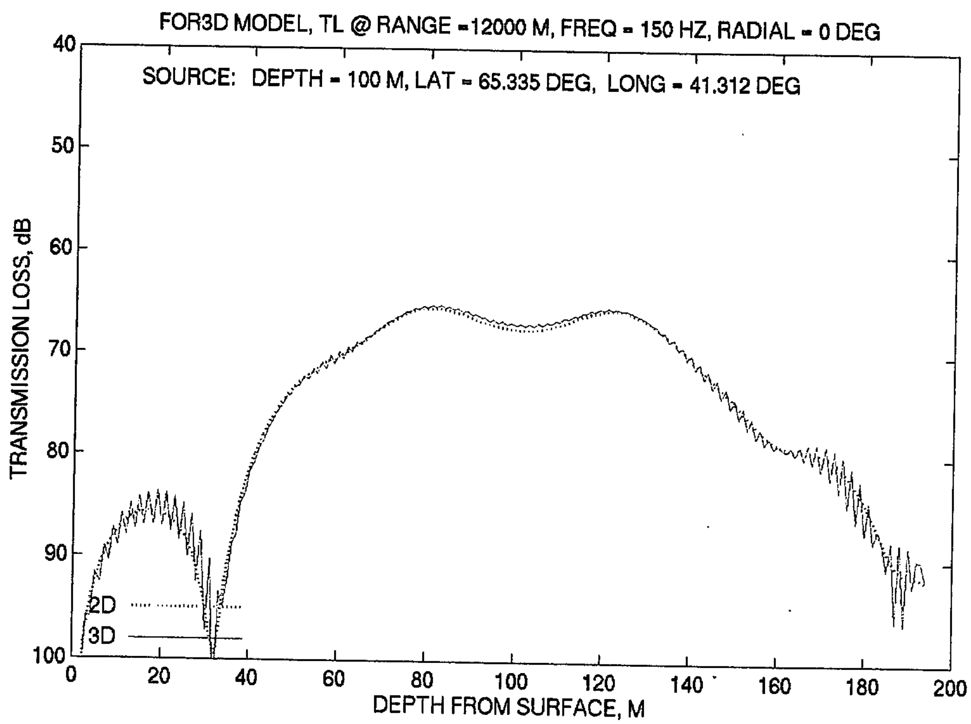


Figure 18

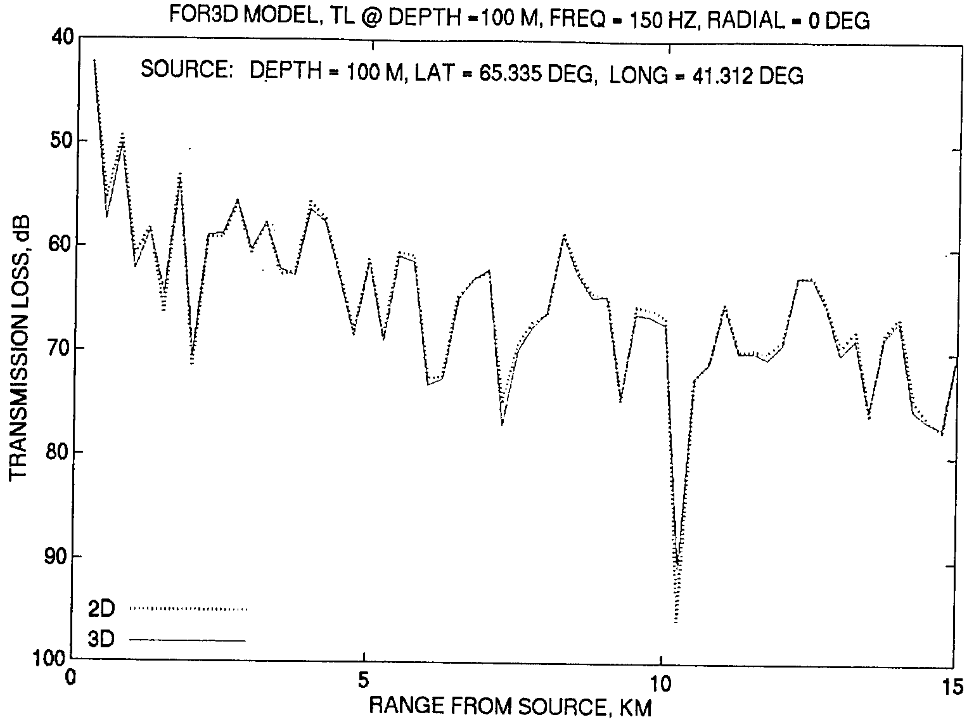
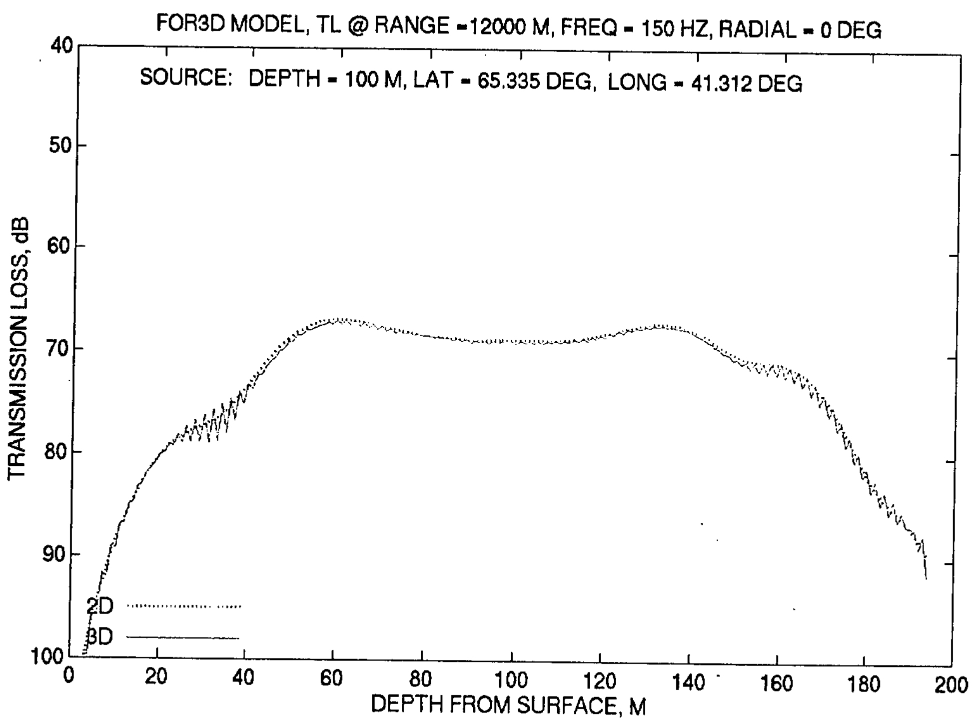


Figure 19



5.3.2 Comparison to Results by the Model Author

In some respects, the results obtained may appear to be in conflict with the result of Lee [7] cited earlier for a 50 hz source, using 1° radial separation. As may be recalled, those calculations were for a sloped bottom condition with propagation tangential to a Gulf Stream eddy (which has similar cross range gradients). The 1° radial separation is equivalent to 8.7 and 58 wavelengths at 15 and 100 km respectively. This is quite different from the azimuth step used in this study.

Either the authors of the Model were:

a. Unconcerned by this large multiple of the wavelength in the azimuth step size,

b. Unaware of any effects caused by this choice, which is rather doubtful,

c. Or simply compromised the test case because of computing resource limitations imposed by use of finer azimuthal grids.

The prediction of up to 8 dB TL differences between the two modes may have been the result of the sloped bottom bathymetry rather than horizontal refractive effects. It would be interesting to know what their result would have been for an otherwise similar case with flat bathymetry.

5.4 Fields with Maximum and Minimum in the Cross Range SV Gradients

Obvious omissions of the study thus far:

1. The previous studies of the Day 8 SV field did not to propagate all the way through the pinched off feature as occurred for the case reported by Lee. The conditions shown in the middle column of Table 1 correspond to the largest problem supported by the computing resources used in this study. The computational wedges presented thus far do not exceed that limitation.

2. With the present computational constraints, the previous SV fields have no features small enough to be covered and traversed that have a maximum or a minimum in the cross range gradient. Regions containing such zones should produce corresponding shadow or focused zones behind the feature, which in turn requires propagation all, or at least nearly all the way through the feature. Differences that are easier to observe and quantify should occur under those conditions.

5.4.1 Day 8, SV Field, Compressed Horizontally by a Factor of 4

To address the limitations above, the SV field of Day 8 was compressed horizontally by a factor of 4. Thus, the pinched off feature (vortex) total dimension is reduced to less than 15km which allows propagation nearly all the

way through the feature without computational compromises. See the earlier description of this field. For convenience, the domain center remains the same as the original field.

The horizontally compressed field is shown in Figs. 20 and 21. In the input stream to the FOR3D Model, this compression has the effect of changing the grid increment of the SV field from 1000 m to 250 m in the X and Y directions. The wedge shown in Fig. 20 displays a few of the propagation tracks. Propagation is nearly all the way through the feature with source at the apex of the wedge. The central radial is aligned approximately with the minimum of the cross range SV gradient.

If the horizontal refraction is sufficiently great, a focused energy zone should exist behind the feature along the central radial. Thus the TL along the center radial should be less for the 3D-mode since it accounts for horizontal refraction.

Figures 22 and 23, dB TL depth versus range plots, are presented for the compressed SV field. As for earlier cases, even critical examination of such plots fails to reveal differences between the 2D and 3D-modes. However, minor differences can be seen in Fig. 24, a horizontal slice, and Figs. 25, 26, and 27 vertical slices at different ranges. The differences, though small (approx. 1 dB), are fairly consistent and in-line with the anticipated convergent lens effect. This would appear to validate that 3D refractive effects are being accounted for by the 3D-mode to a greater extent than by the 2D-mode.

5.4.2 <u>Day 8, SV Field, Compressed Horizontally but with an Inverted Gradient</u>

Details of this field's generation were presented earlier. The compressed and inverted field is shown in Figs. 28 and 29. In this case alignment is along a maximum rather than a minimum. Behind the feature a shadow zone should exist, thus, the 3D-mode might be expected to produce a greater dB TL.

As before, differences between dB TL depth versus range plots, are not noteworthy, therefore these plots are omitted. However, minor differences can be seen in Fig. 30, a horizontal slice, and Figs. 31, 32, and 33 vertical slices at different ranges. The differences are again very small (approx. 1 dB). However, in this case they are inconsistent, i. e., sometimes higher, sometimes lower. Even though this is contrary to the anticipated divergent lens effect the 3D refractive effects in this case may be sufficiently small to simply place the differences down into the noise level.

In the case of a convergent lens zone, greater energy should arrive at the focal axis of the zone as was observed. On-the-other-hand, it is possible that little effect would occur at the neutral axis of a mildly divergent zone, simply because

Figure 20

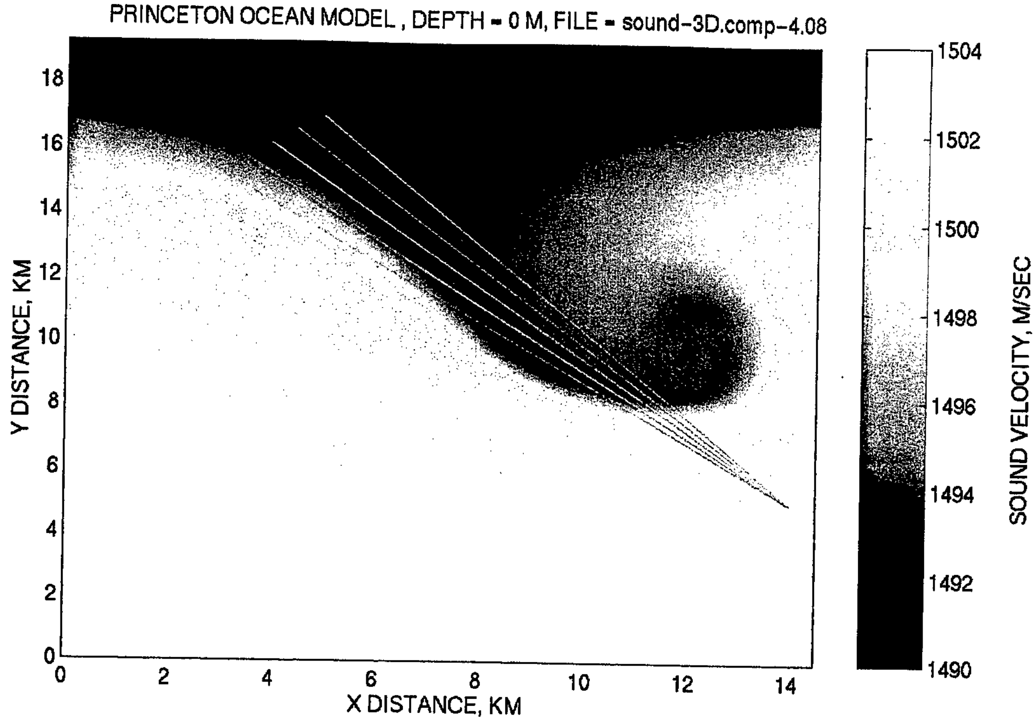


Figure 21

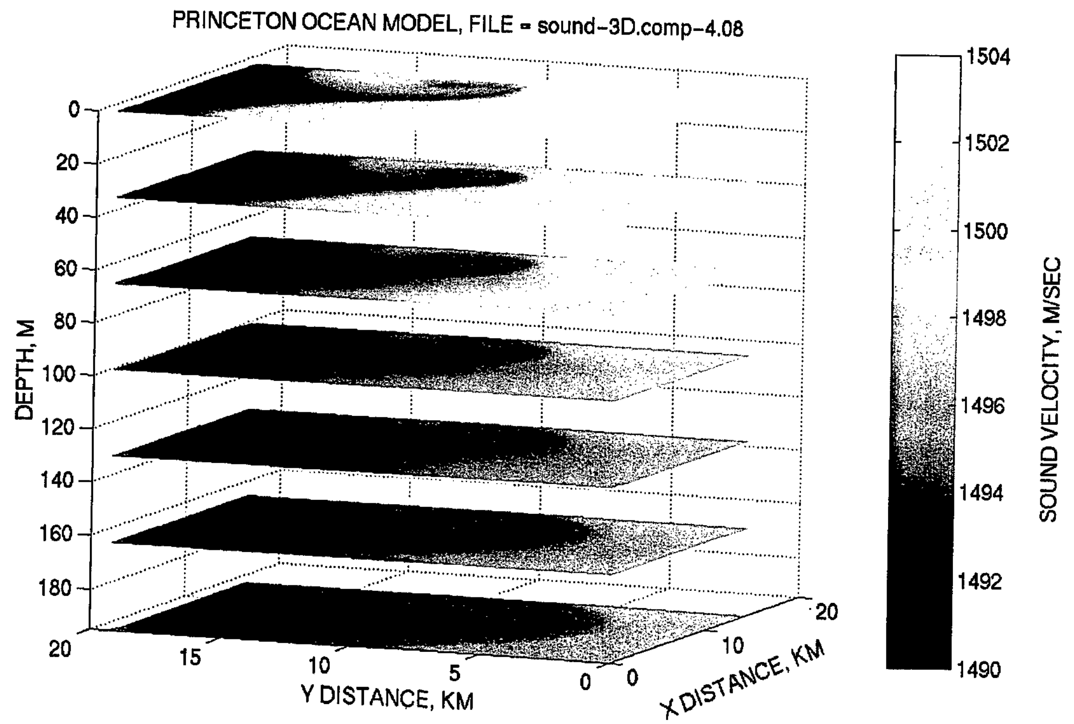


Figure 22

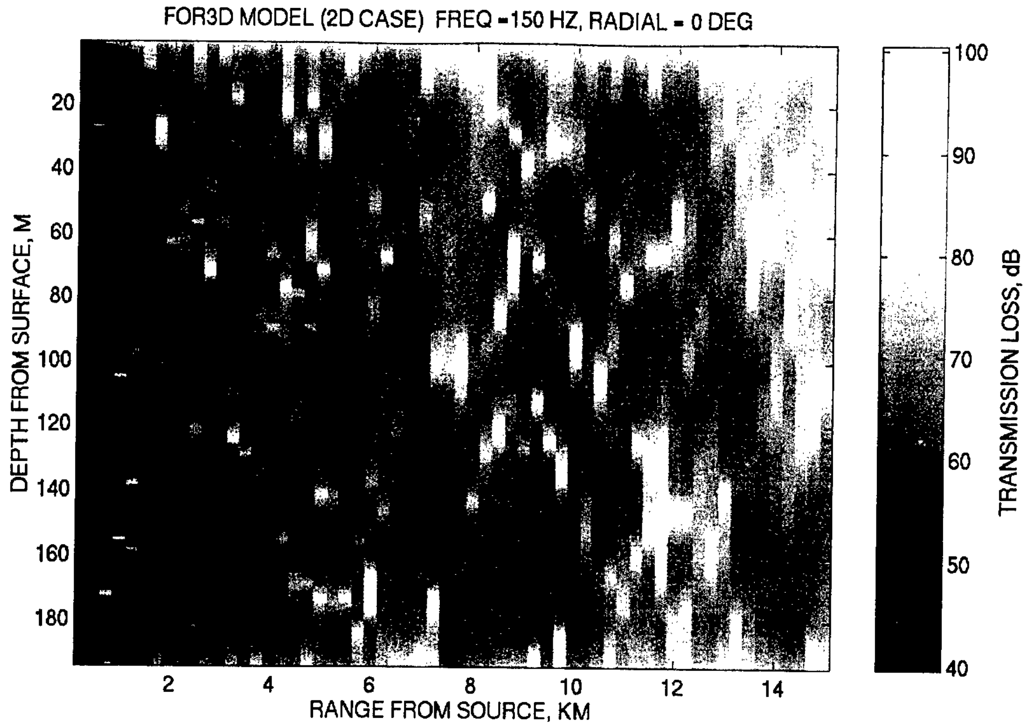


Figure 23

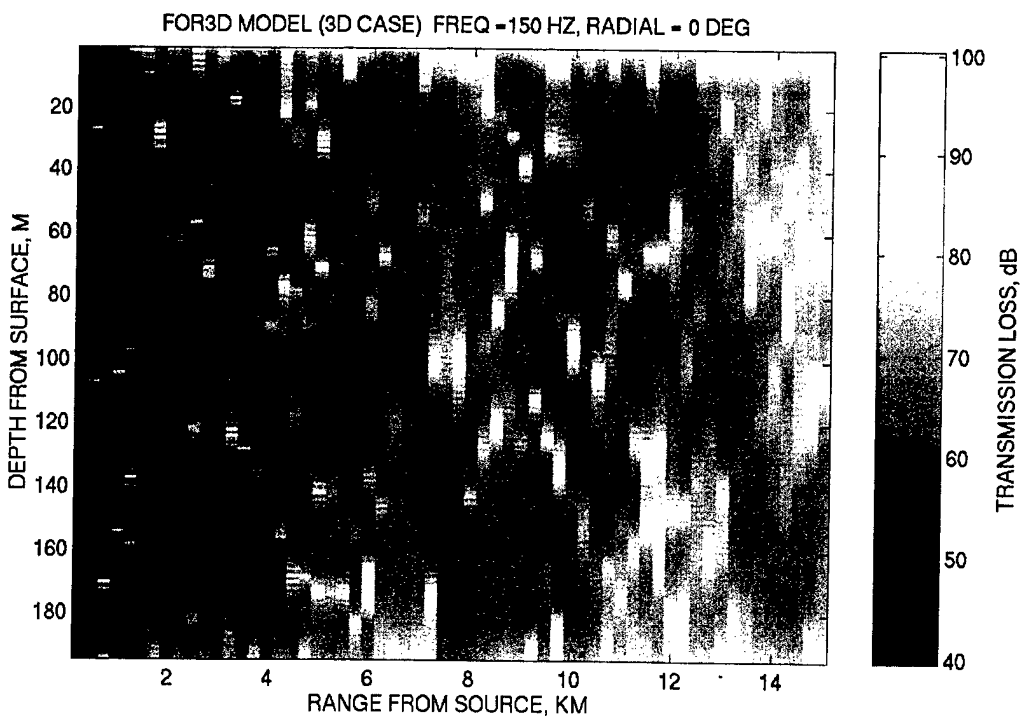


Figure 24

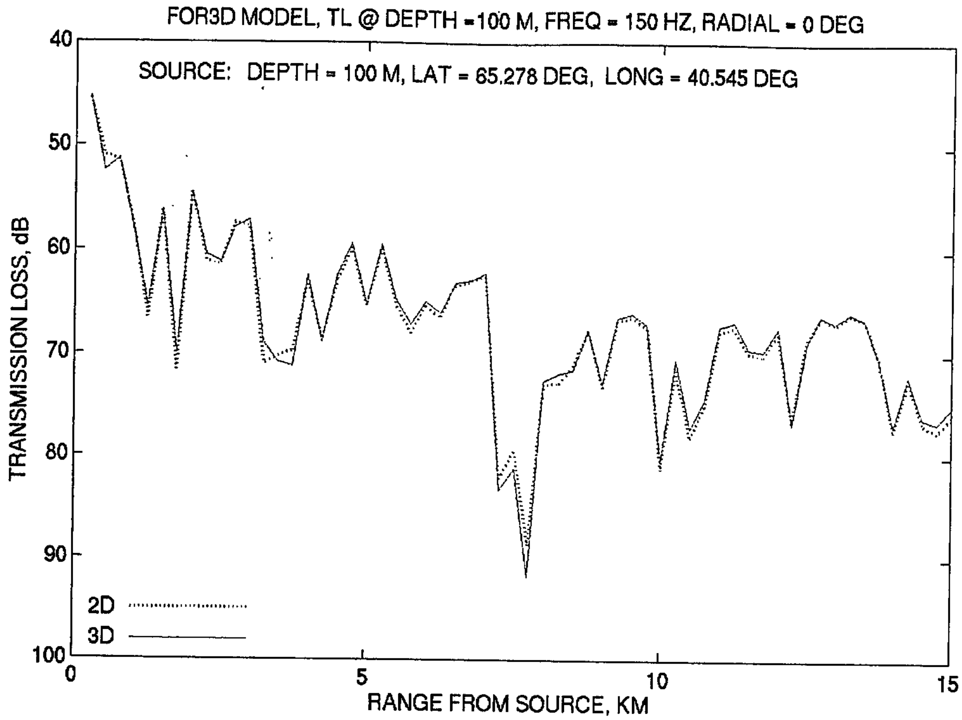


Figure 25

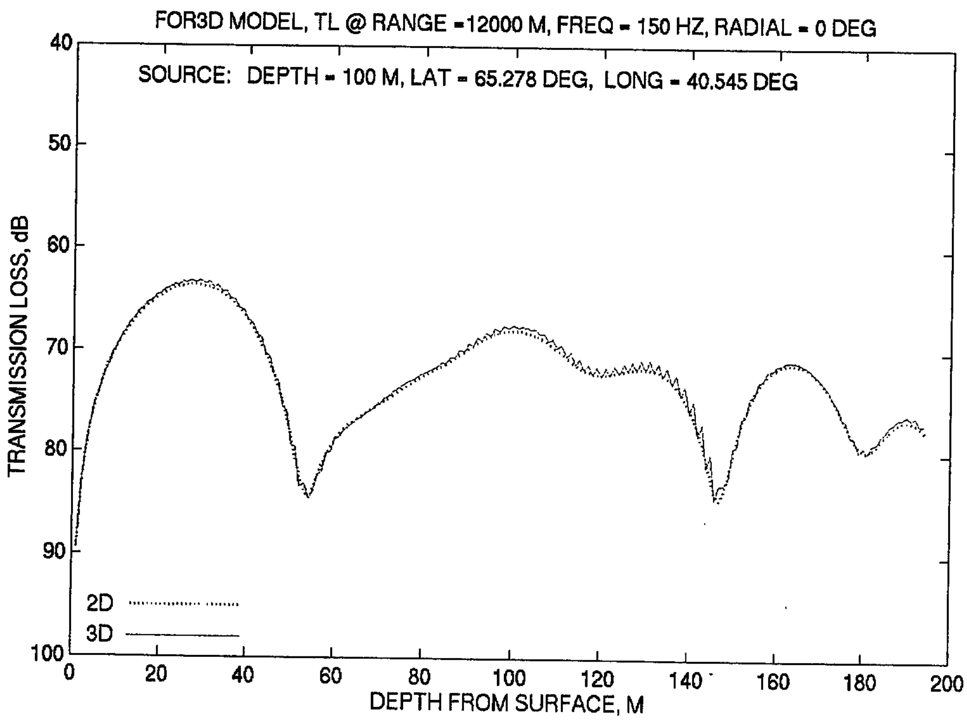


Figure 26

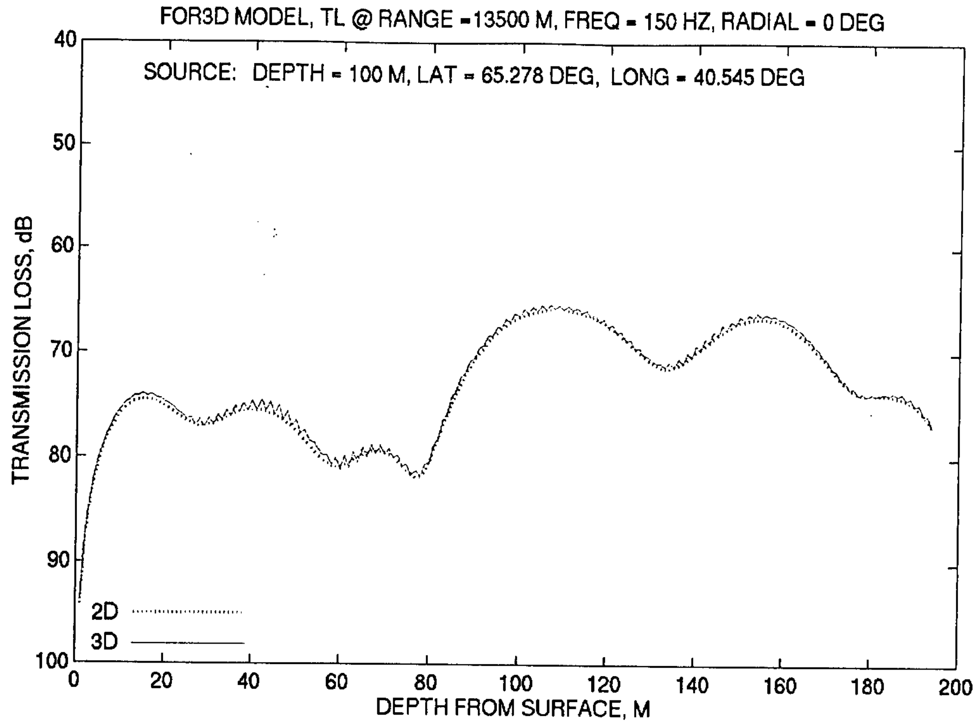


Figure 27

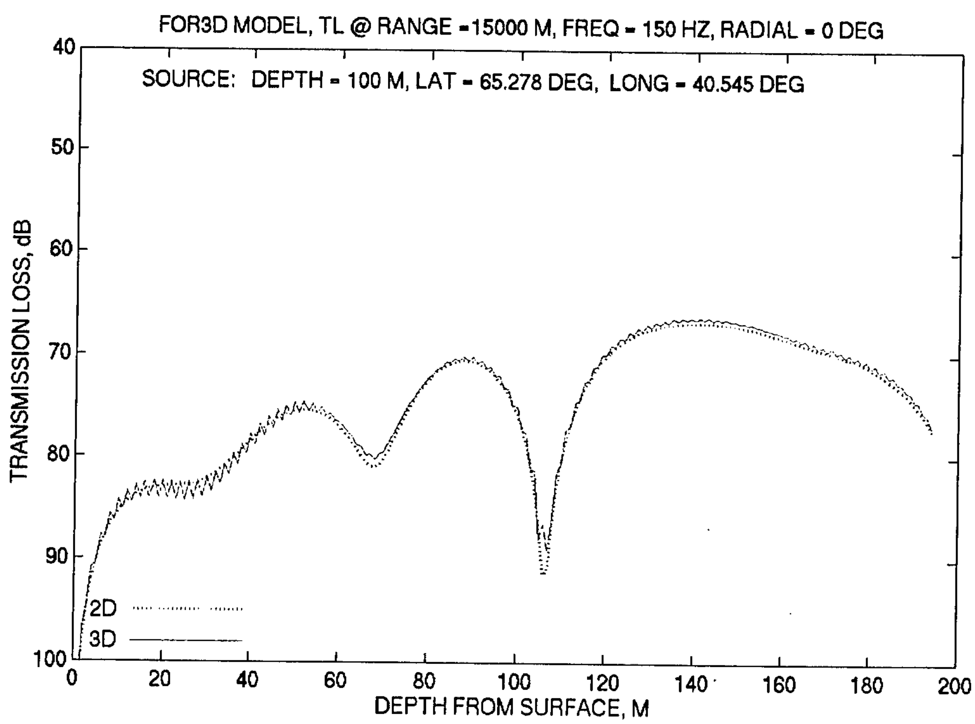


Figure 28

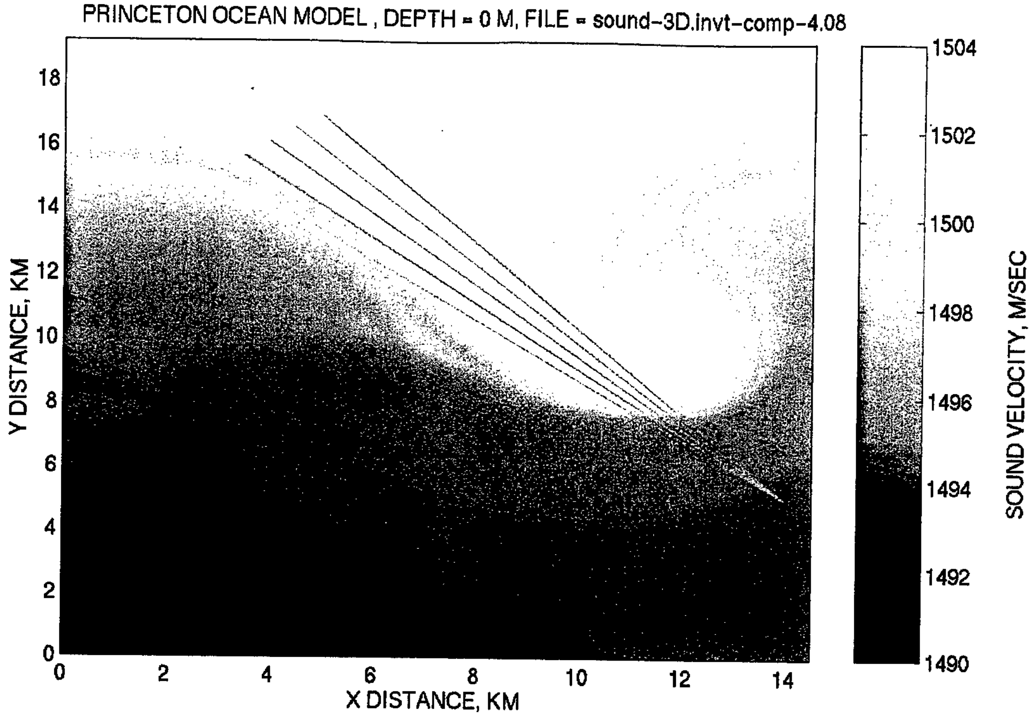


Figure 29

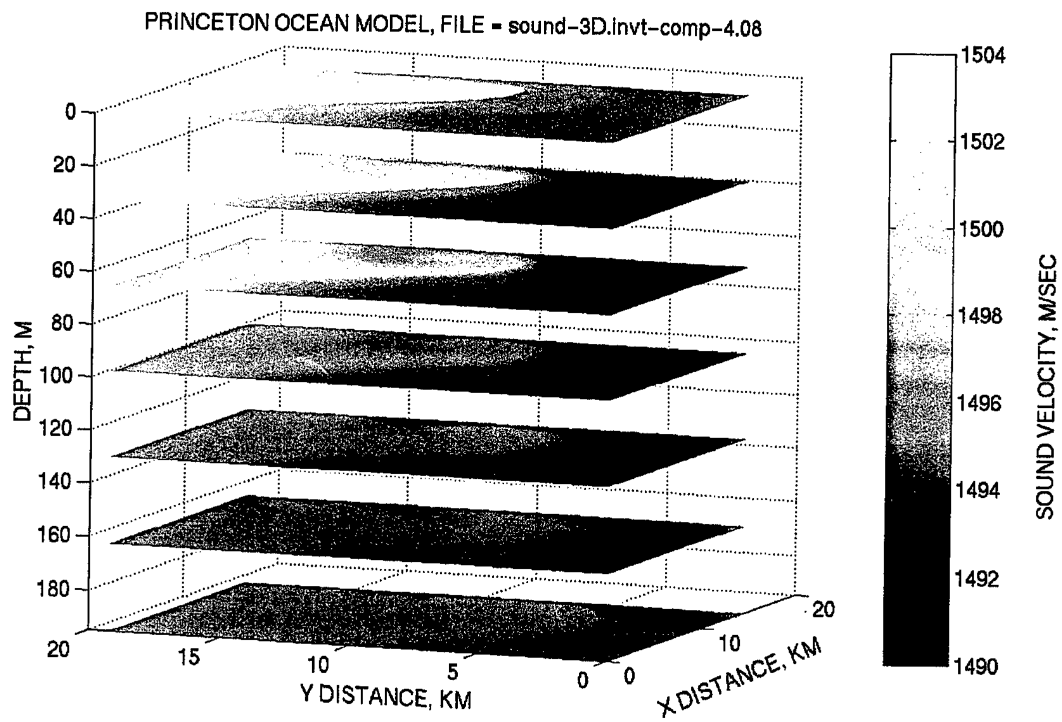


Figure 30

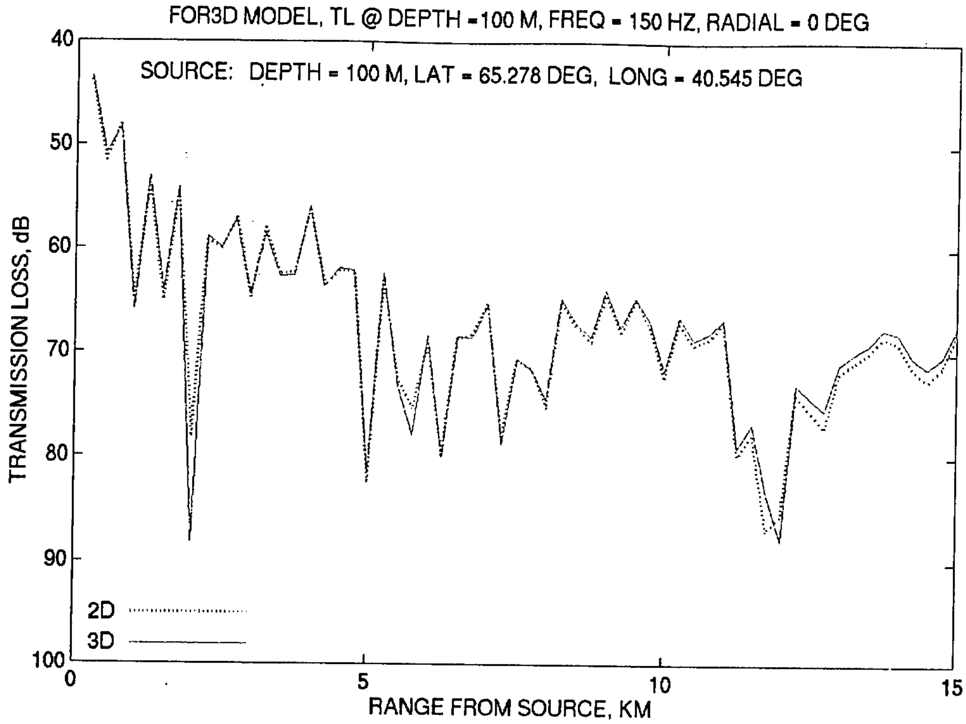


Figure 31

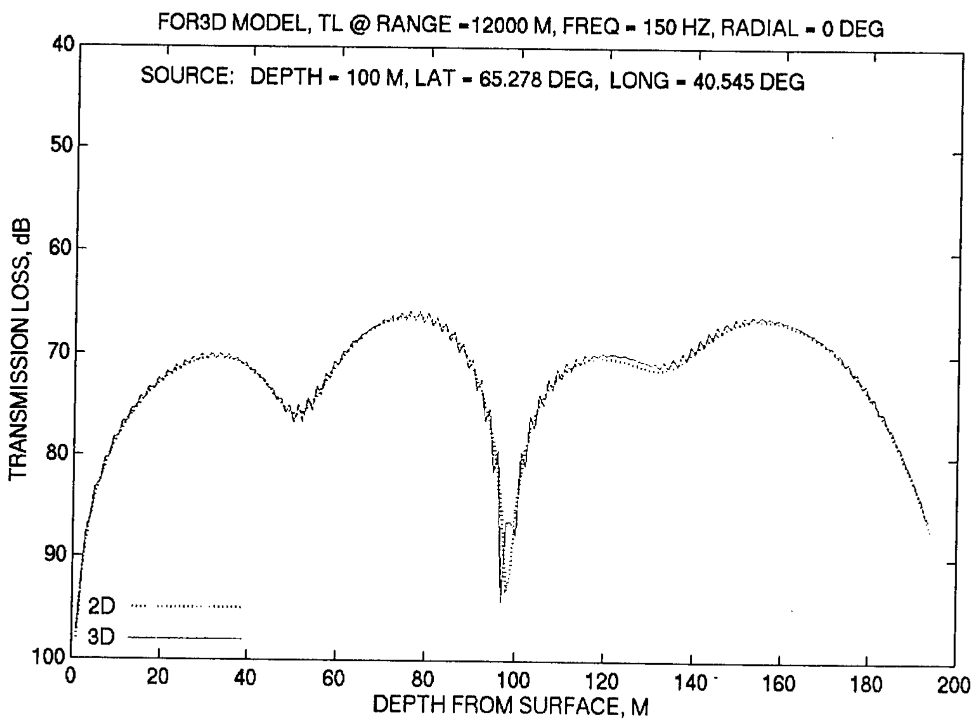


Figure 32

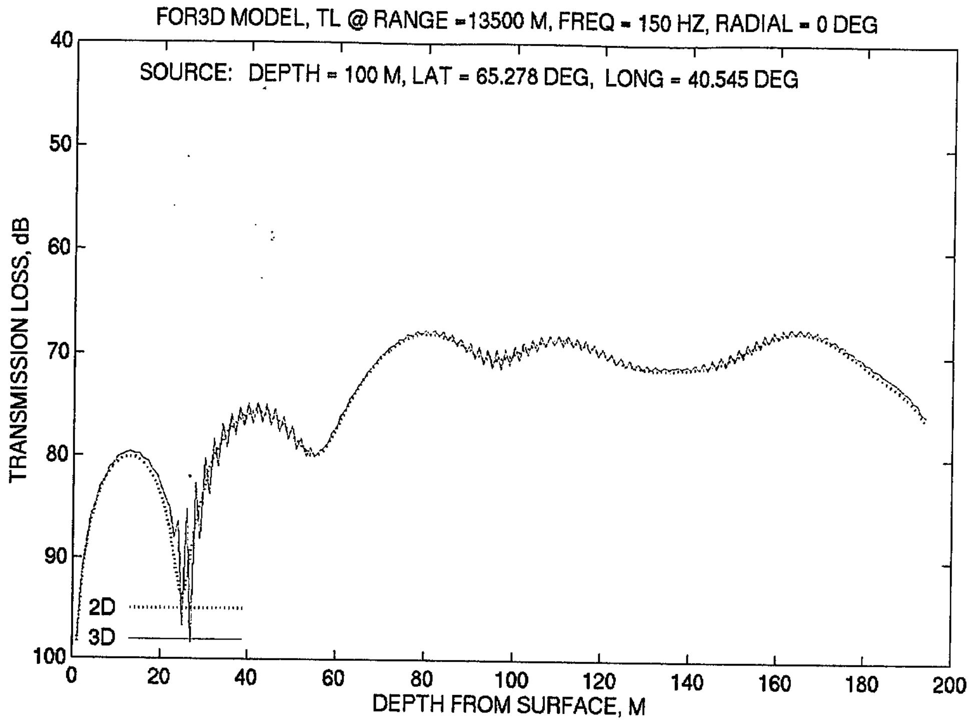
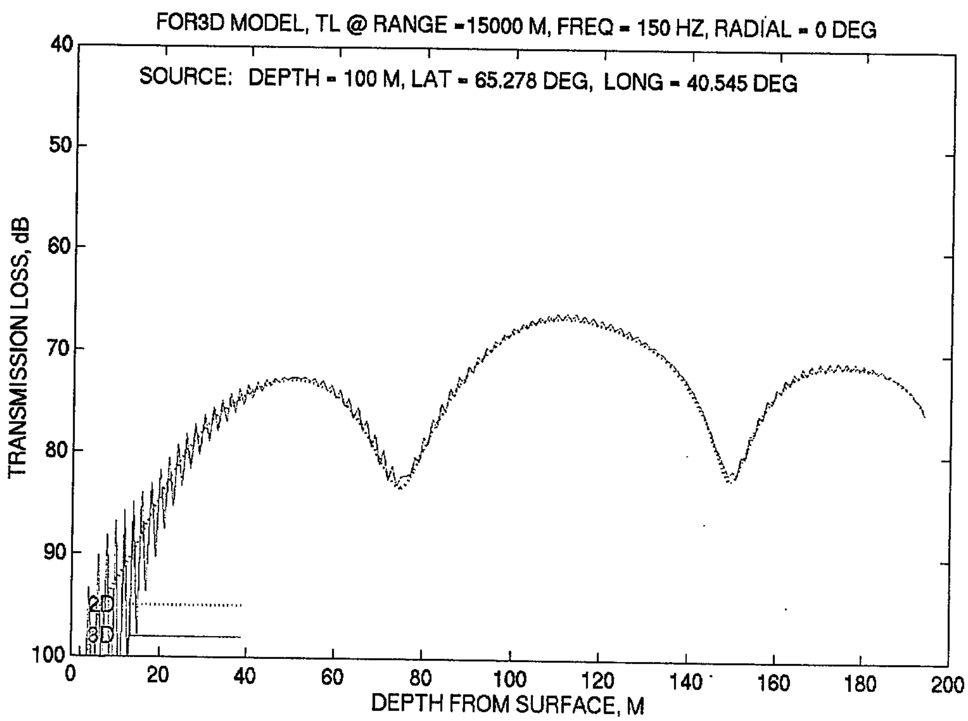


Figure 33



along that axis no refraction occurs at all. In any case, the 2D-mode prediction is for practical purposes identical to the 3D-mode prediction.

5.5 Discussion of Cases having a Minimum or Maximum Cross Range Gradient

As noted above, the anticipated convergent effect on dB TL differences occurred for the case with minimum cross range gradients. Above 12km range, the effect was at least 0.5 dB. Even though this difference is meager, it does appear to correspond to the horizontally refractive effect of the horizontal SV gradients. That the converse was not observed for the maximum zone was perhaps disappointing, as it did not validate an improved refractive effect of the 3D-mode. A careful review of the SV field shown in Fig. 28 reveals that the gradient is fairly flat along the central axis, probably accounting for the small effect observed.

One of the concerns in using the Model in a shallow water environment is that if too low a frequency is used, vertical feature discrimination may not be possible. Thus, the challenge has been to develop study cases to determine efficacy of the Model without exceeding present computational resources. Note that during the compression of the Day 8 sound field, the vertical gradients were not changed whereas the horizontal gradients were increased by a factor of four. Since the vertical gradients were unchanged, vertical refractive effects should have been unaffected. On the other hand, the horizontal refraction should be increased by the factor of four.

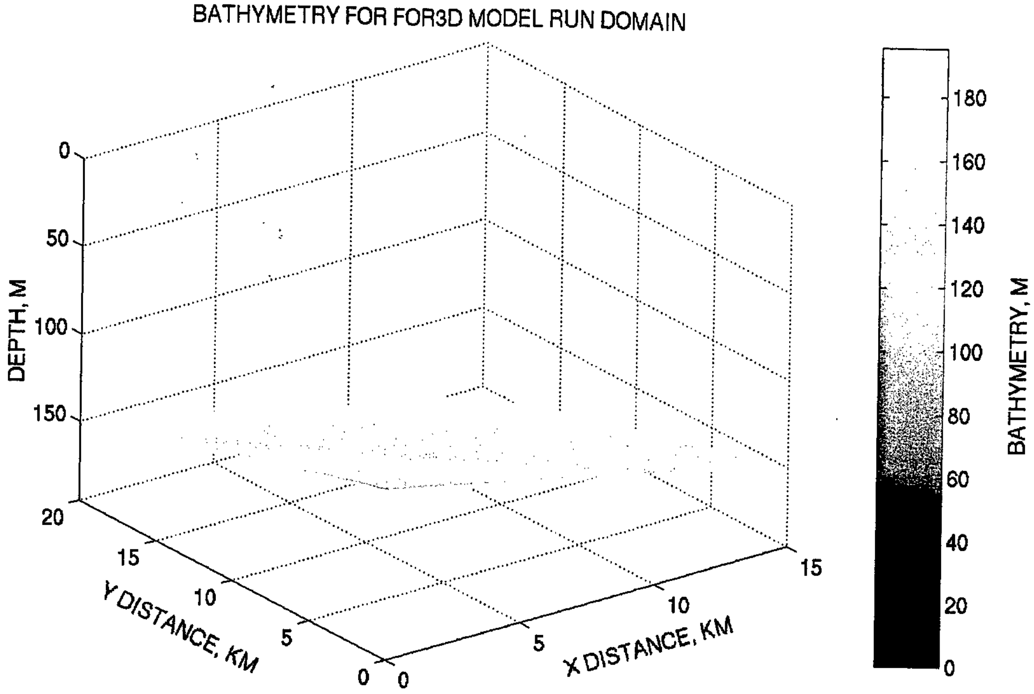
Without the current computational limitations, the original (uncompressed) sound field could have been studied along an analogous track to 60 km instead of 15 km. Noting that refraction, range and gradient curvature effects are interchangeable and that there are no reflecting boundaries horizontally, the lens effect produced at 60 km for the original field must be the same as for the compressed field at 15km.

Though the coastal regions of interest do not have SV gradients of the magnitude of the compressed fields just presented, the use of this compression technique has allowed determination of the magnitude of the lens effects from horizontal refraction at 60km for the Day 08 feature.

6. EFFECT OF A SLOPED BOTTOM

All studies presented thus far were for flat bathymetry. The FOR3D Model has a variable bottom bathymetry capability. A single test was made using the SV field shown in Figs. 20 and 21 (The Day 8 field compressed by a factor of 4) and propagation track shown in Fig. 20 for a sloping bathymetry. The sloping bathymetry used is shown in Fig. 34. Study of these figures reveals that the central propagation track is along an approximately constant depth of 150 m. The slope gradient is largely in the cross range direction.

Figure 34



The results obtained are shown in Figs. 35 and 36 where the effect of shallower bottom is quite obvious. However, there is very little difference between the 2D and 3D-mode plots. Overlays of the TL at iso-range and iso-depth as presented for the earlier studies showed that difference in dB TL were less than 3 dB. The 3D TL was consistently less than the 2D TL though the effect was quite small. This small difference could probably have been anticipated, as out of vertical plane reflections can travel horizontally only small distances before they are substantially absorbed by bottom interaction when there is a shallow bottom.

For the purposes of this study (primarily concerned with flat bathymetry) these results are sufficient to show that gently sloping bottoms in relatively shallow regions can be modeled successfully by either of the two computational modes.

7. EXAMINATION OF MODEL 3D COMPLEX PRESSURE FIELD RESULTS

As 3D complex pressure fields are extremely difficult to interpret, it is conventional (as done thus far) to plot these results as dB TL. On such plots, the main visual impact results from the phase interaction of sounds arriving via different paths. As long as the different modes produce 3D fields having virtually identical relative phases (i. e., phase differences modulo 360 relative to the same vertical reference point) the interference patterns produced will be nearly identical. Small magnitude differences are visually swamped by the resulting

Figure 35

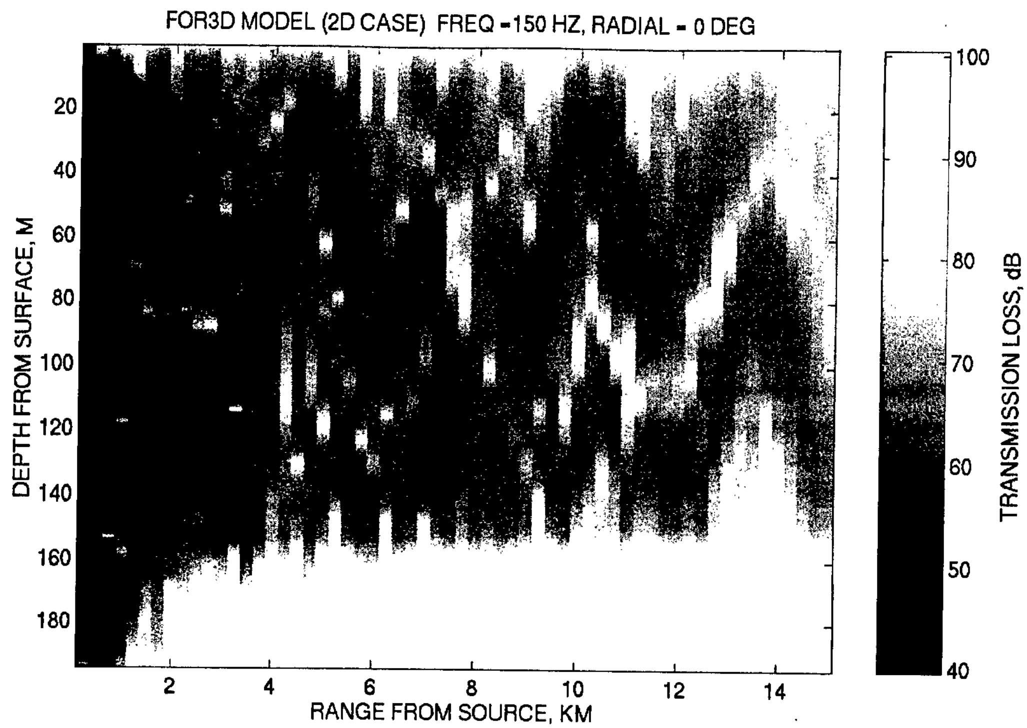
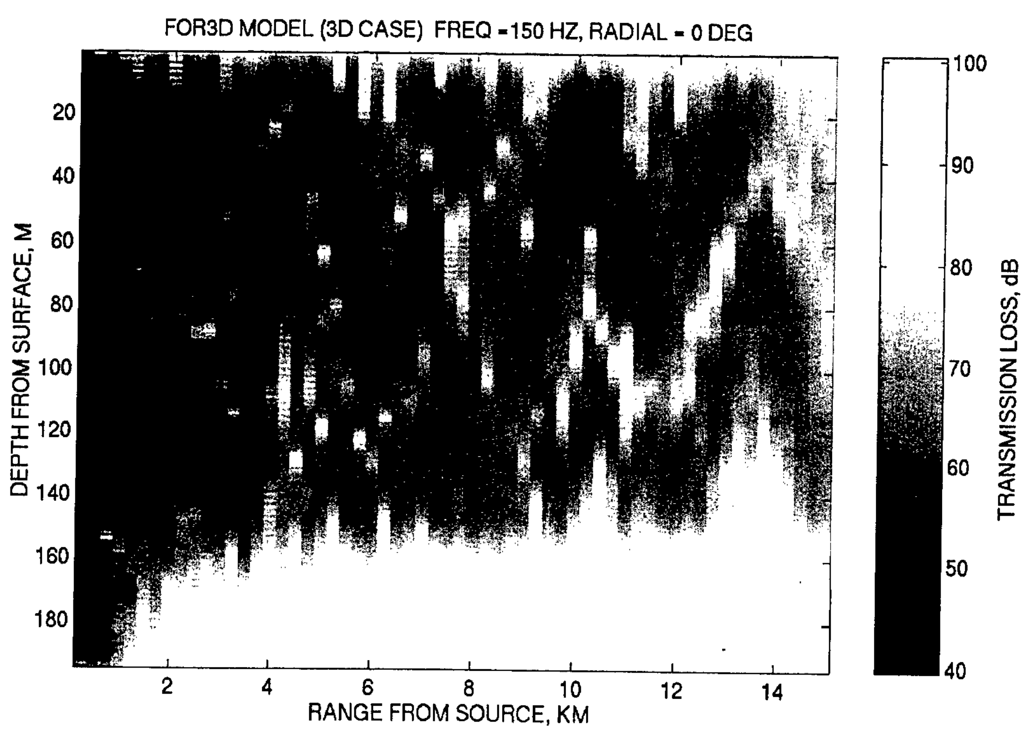


Figure 36



interference patterns. That this has occurred can be verified by comparison of Fig. 9 with Fig. 10 and Fig. 22 with Fig. 23.

The dB TL (on the average) resulting from energy dissipation is systematic and gradual in range, except for TL from interactions at the bottom and surface or other boundaries. Little difference in dB TL from sound travelling in a curved (horizontally refracted) path compared to a straight path should result from dissipative effects when there are no boundaries since the path length differences are quite small (Note that: sin(3)=tan(3) to about 0.1% accuracy.)

7.1 Additional Information Available from the FOR3D Model

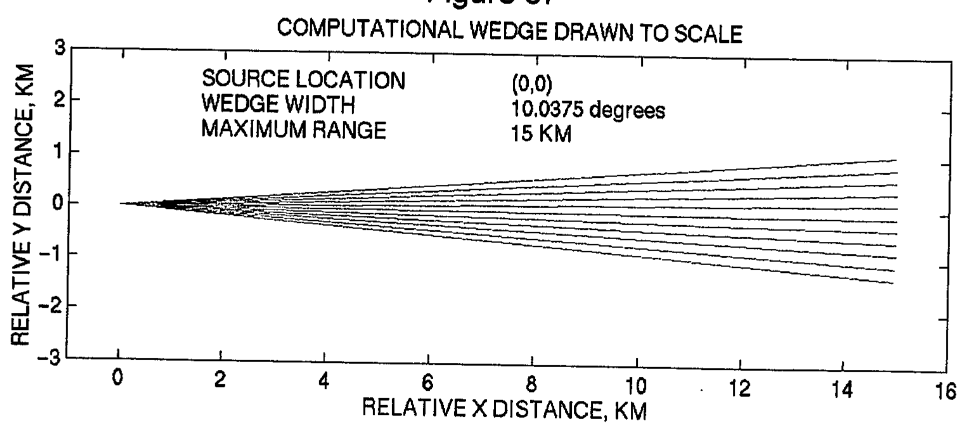
The computational wedge that has been used in earlier studies is shown to scale in Fig. 37 with most of the radials omitted. It is obvious that the end of the tracks form a horizontally circular pattern. The complex fields determined for the two computational modes at the end of these tracks are available for comparison.

A case originally presented (Fig. 6.) in the context of dB TL was for the SV field with a cross range gradient of -0.01 sec^{-1}. The computational wedge was located as shown in Fig. 13. As the SVs are lowest on the left side of the wedge (viewed in the propagation direction), sound rays would be curved from right to left. Recall that there were no significant dB TL differences, Figs. 18 and 19.

7.1.1 <u>Magnitude</u>

The relative magnitude results at the end of all (2751) tracks are shown in Figs. 38 and 41 for the 2D and 3D-modes respectively.

Figure 37



In Fig. 38, magnitude systematically increases from left to right, whereas in Fig. 41, there is a surplus at the left and deficit at the right. In 3D-mode, the 2D radial solutions are imposed at the left and right boundaries. One way of viewing this is that energy arriving from outside the wedge is blocked from entry and energy leaving the wedge is trapped inside. Since ray curvature is from left to right, there

should be energy entering on the right side (from the adjacent region) and leaving on the left side. Interpretation of the 3D result above is consistent with that explanation.

It is reasonable to discard results near the wedge boundaries. Once this is done, magnitude agreement between the 2D and 3D-modes is virtually identical.

7.1.2 Phase -- Comparison

The phase information can be viewed in two ways (1) a simple comparison of the relative phases and (2) by treating the values as input to a horizontal hydrophone array, processed by an FFT beamformer (a common technique).

In accordance with (1) above, Figs. 39 and 42 present the phase information for the 15km terminus of the radials at the 100m depth. Note the orderly shift in phase from left to right in both of these figures. Again, using only the heart cut radials, in this instance, those between –3 and +3 degrees, the phase shift structures are virtually identical. Though the specific phases are not identical in the heart cut, the phase shift relative to any particular radial is the same for the 2 modes. It is reasonable to assume that kind of result occurs at levels than the source level as well. Notably, energy distribution from the Green's starter function is not confined to the horizontal plane of the source. In spite of this, it is apparent that the relative phases must have been nearly the same for the 2D and 3D-modes, to wit the nearly identical results (interference patterns) obtained in Figs. 18 and 19.

7.1.3 Phase – Beamforming

The complex SV values at the ends of the radials when processed by a conventional double FFT beamformer using a Hanning window produce the results presented in Figs. 40 and 43 for the 2D and 3D-modes respectively. The Hanning window has the effect of emphasizing the heart cut values. *Since the FOR3D Model produces results for the single input frequency, the SV field at the ends of the radials is analogous to the output from the first stage of the double FFT beamformer and thus may be used directly.*

The beamformed results for the central beam in the absence of horizontal refraction would be the broadside beam, i. e., the beam located at angle 0. The apparent direction of sound arrival is about -3.2° for both the 2D and 3D-mode results. Clearly the 2D-mode as well as the 3D-mode has determined the same horizontal acoustic refraction. When listening to hydrophones or viewing the output from a Model such as FOR3D, all that can be detected is the phase, modulo 360. Thus the absolute phase information relative to the time origin of the wave front is unavailable and cannot be verified experimentally, all that can be determined is the phase relative to one of the phones in the receiver array.

Figure 38

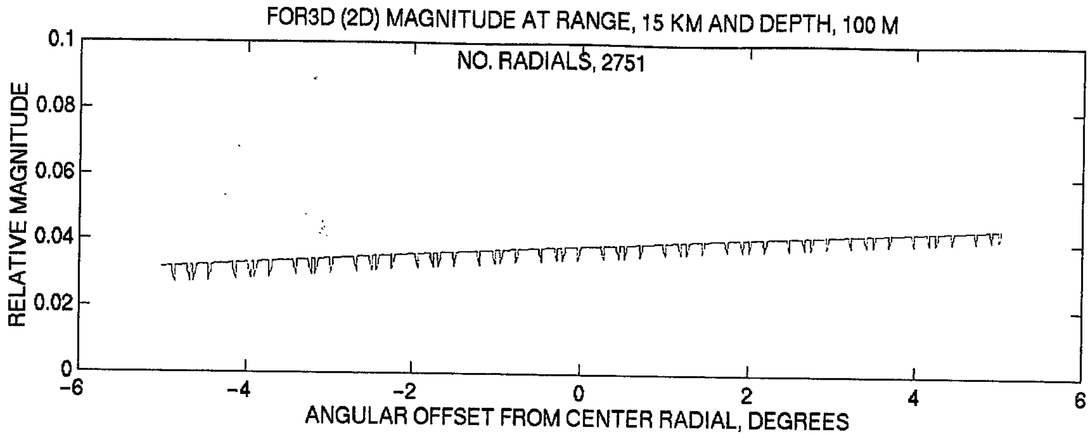


Figure 39

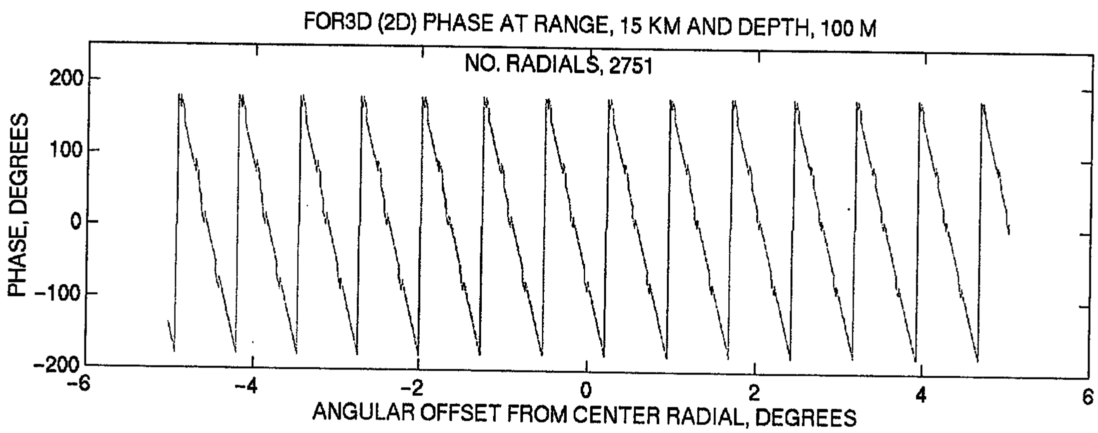


Figure 40

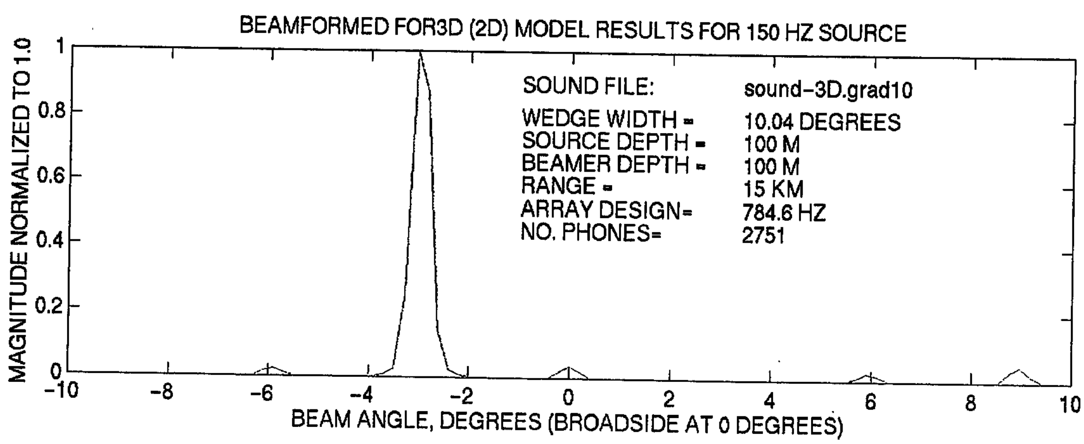


Figure 41

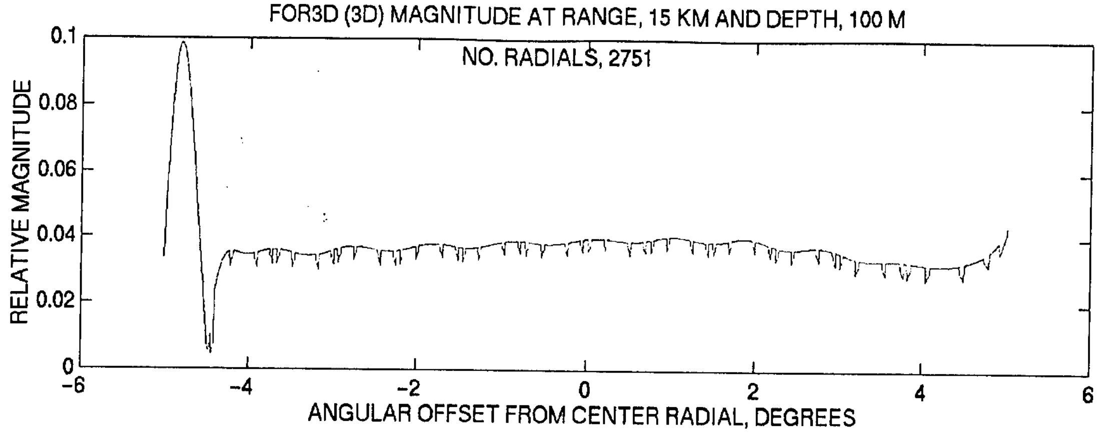


Figure 42

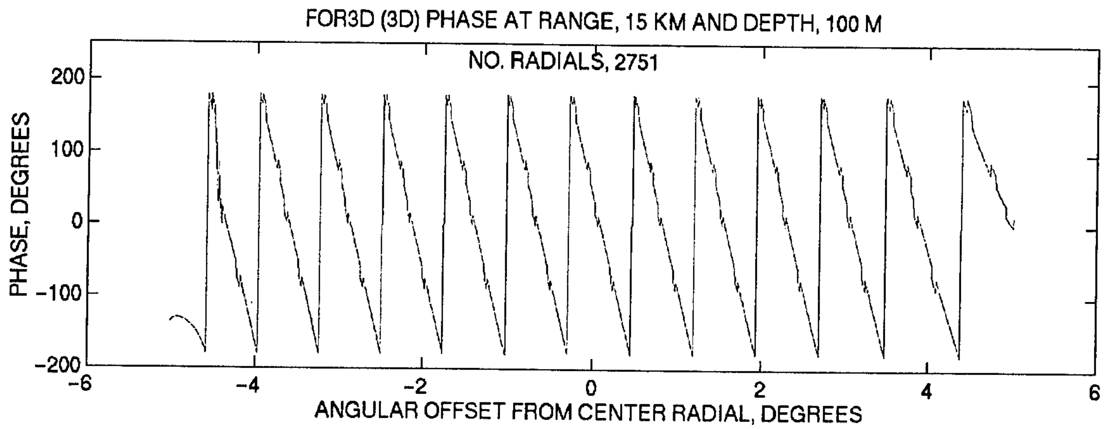
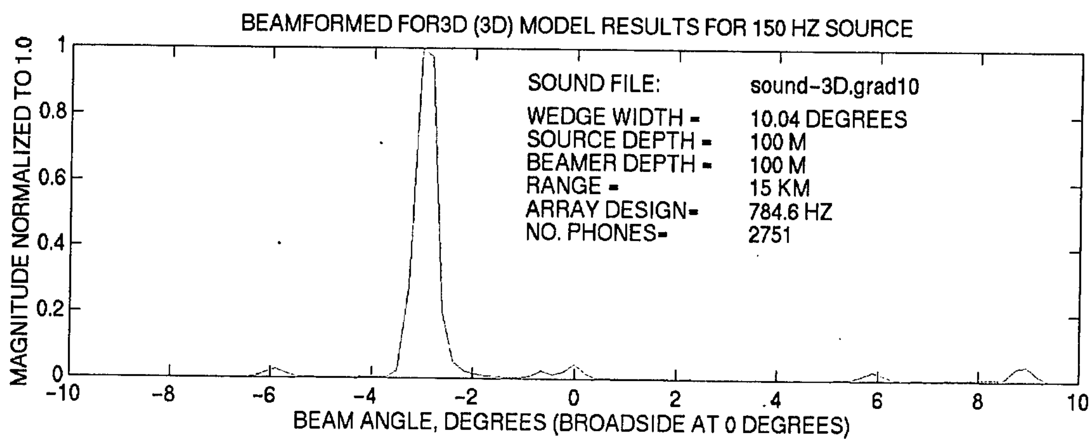


Figure 43



As the beamformed results were virtually identical, the relative phase shifts must have been virtually the same as is evident in Figs. 39 and 42.

8. THE EXTENT OF HORIZONTAL REFRACTION FOR 2D AND 3D MODES INDEPENDENTLY DETERMINED FOR IDEALIZED CASES

A 2D-mode PE model sets all azimuthal environmental derivatives to zero which appears to requires sound to be confined to the calculation plane. A reasonable question is, "How is the close agreement in horizontal refraction, evidenced by the beamforming results above achieved?" Two idealized cases will be presented that can be solved rigorously (the 3D-mode) and non-rigorously (the 2D-mode). The environments of interest to this study have relatively weak horizontal refraction and flat bathymetry. Under those conditions the 2D-mode results, for practical purposes, are shown to be virtually identical to the 3D-mode ones.

8.1 Horizontal Refraction by a 2D-Mode Calculation

A heuristic argument is presented to demonstrate that a 2D-mode acoustic model should account for most of the horizontal refraction caused by the medium. Acoustic models such as FOR3D attempt to solve the 3-dimensional wave equation, producing a 3D complex pressure field for a defined computational region and starting field. In the case of the FOR3D Model, two computational modes are supported, the 2D and 3D-modes. The 2D-mode of the FOR3D Model is similar to other models where the solution is limited to a vertical slice. Even though the terminology 2D-mode may not be mentioned, it is implicit for such methods.

8.1.1 <u>The 2D-Mode Tacit Assumption</u>

Imagine an environment where all environmental properties for a particular depth and range are independent of azimuth from source, i. e., circularly symmetric about the source. If the source was also circularly symmetric, any solution must be identical for every vertical plane passing though the source center. Each of these solutions could be viewed as a 2D-mode solution to the problem. What is less obvious is that every 2D-mode solution has tacitly evoked the above condition during its computation as all environmental derivatives in azimuth have been set to zero. A consequence of this tacit assumption is that all wavefronts are presumed to be horizontally normal to the slice during the computation. This is the same assumption invoked for 2D ray trace Models, i. e., all rays are confined to the slice.

During the 2D-mode calculation, the actual environment in the azimuth direction must be determined, which in general is azimuth dependent. What is being pointed out here is not in conflict with that reality, only that the 2D-mode calculation invokes this assumption locally during each computation wherein the environmental data are updated for each new azimuth.

8.1.2 An Idealized Case Where the Exact Solution Is Known

In the previous paragraph, a basis has been laid for the following example. First define an idealized, though unrealistic, 3-dimensional SV field which can be analyzed simply. Assume a SV field that is constant vertically, having only a SV change at a single boundary. Figure 44 is shown for a horizontal slice through such a field. The SV boundary is located at Y=15 km. The SVs Cd = 1.55 km/sec for Y > 15.0 km and Ca =1.45 km for Y<15 km.

Plane-wave fronts (for 150 Hz waves), the solid lines, arrive at the boundary with an incident angle of 30°. To avoid congestion, only every 1000th wavefront is shown. Above the boundary the refracted waves continue at angles determined by Snell's Law.

8.1.3 The 2D vs. 3D-Mode Analogy

Noted at the top of Fig. 44 is the 3D-mode refraction (the horizontal one) determined by Snell's Law. The value is accurate for this simple 3-dimensional environment. By analogy, the 3D-mode refraction label seems appropriate since it is the intent of the acoustic model to determine the horizontal refraction correctly (Actually, the 3-dimensional complex field from which such information can be determined).

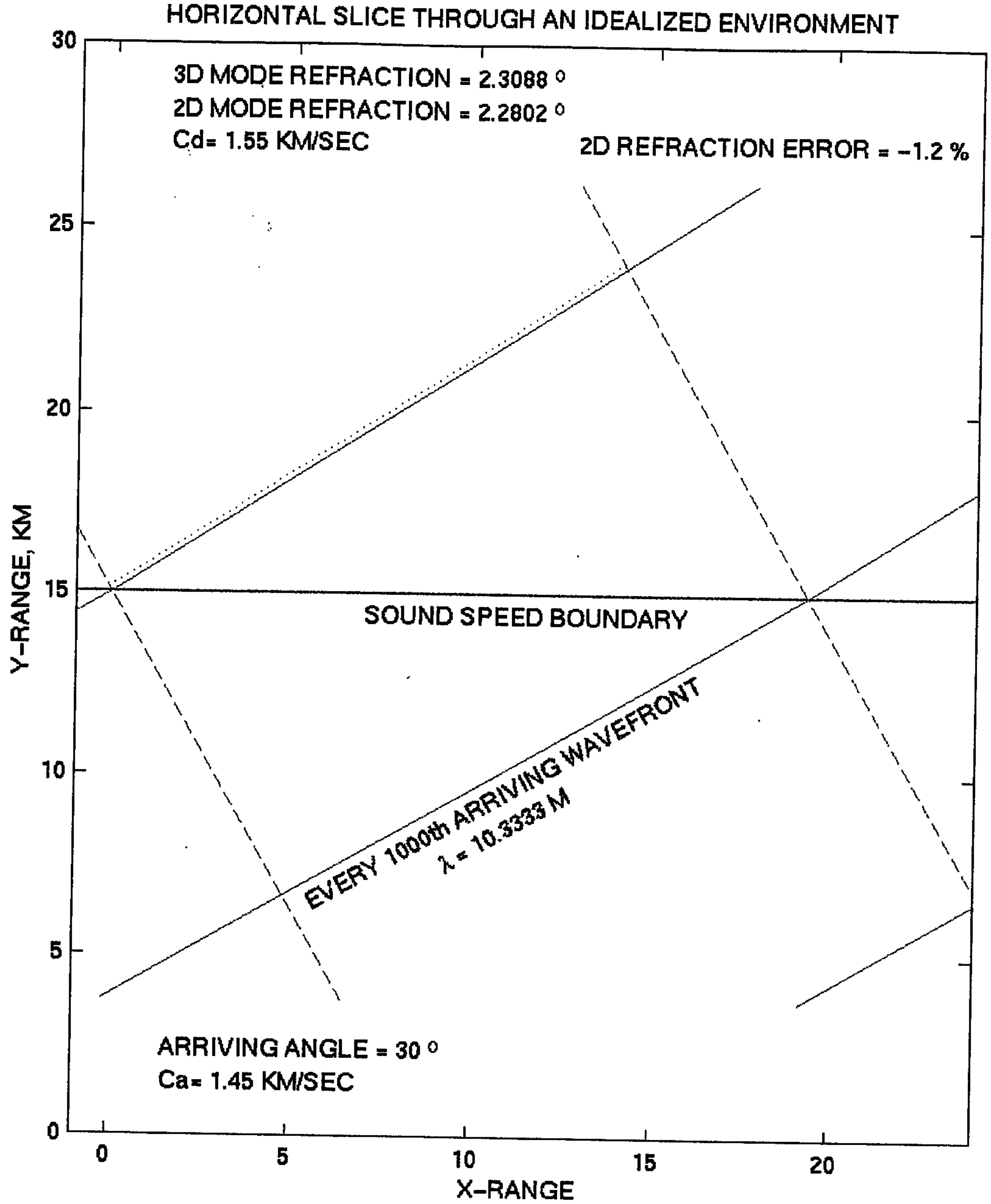
Although the technique has not been explained so far, the 2D-mode refraction is shown also. The 2D-mode refraction is determined by calculations confined to the plane normal to the direction of the arriving wavefronts (as occurs in the 2D-mode acoustic Model). Note that the 2D refraction error = -1.2%. It is clear that the 2D-mode method has accounted for most of the horizontal refraction actually occurring.

8.1.4 The 2D-Mode Calculation Method

Computational models in 2D-mode confine all calculations to a single vertical plane appearing as a straight line in Fig.44. In the figure, the dashed lines are normal to the arriving wave fronts and would represent just the plane implied by the 2D-mode tacit assumption. A simple method, the 2D-mode calculation, for this special case (not Snell's Law) can approximately determine location of the departing wave fronts for fields with relatively weak refraction.

To locate a wave front by the 2D-mode method, points on the wavefront are determined as the distance along any two of the arriving wavefront normals from the SV boundary to the particular wavefront. The location of the wavefront is a straight line extending through these two points. Once the wavefront location is

Figure 44

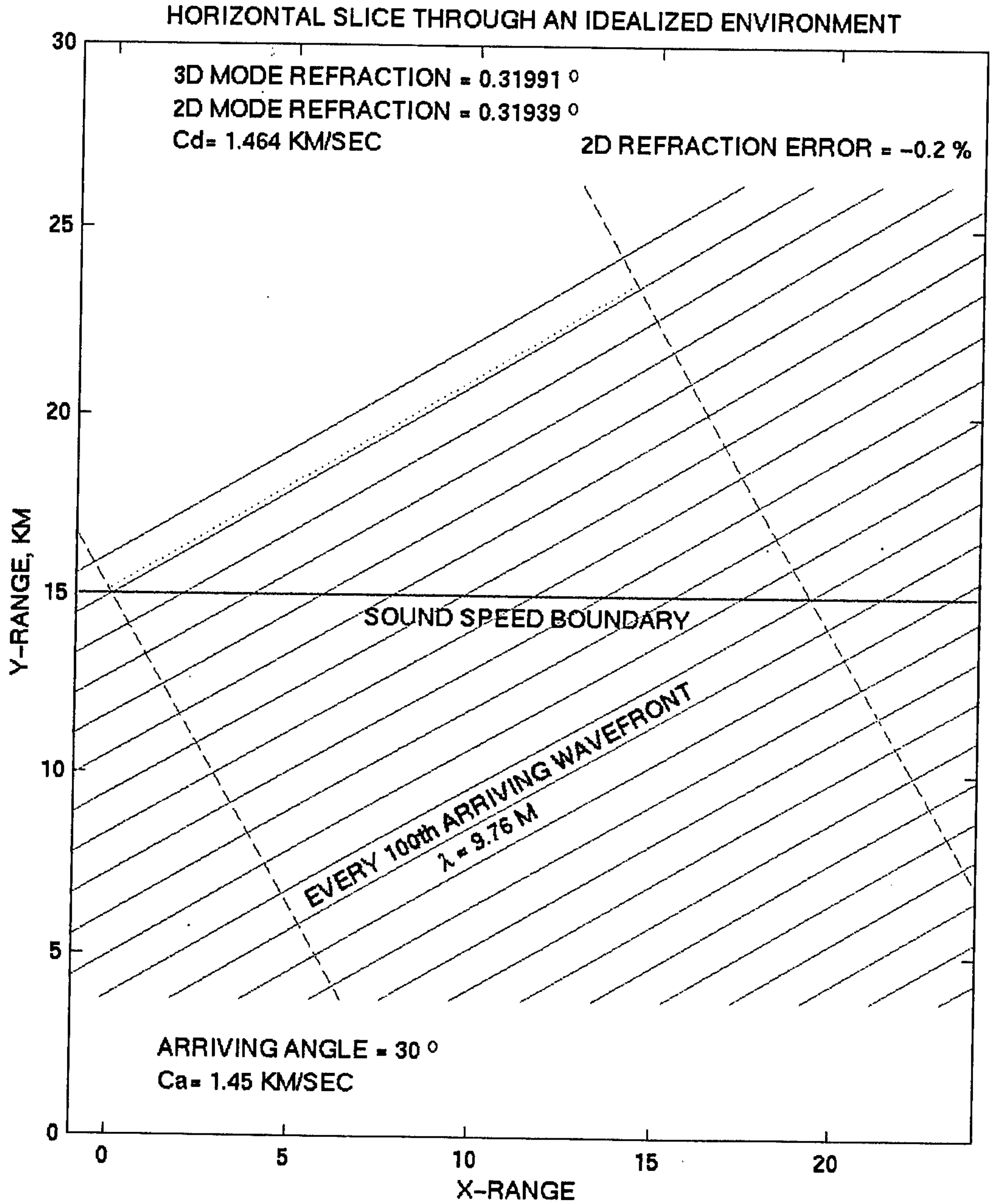


known, the refraction angle can be determined easily. The necessary distances are determined by multiplying the wavelength by the number of wavelength offsets from the SV boundary.

For example, in Fig. 44, the ends of the dotted line correspond to offsets of 15 and 1015 wavelengths at the left and right ends respectively. The offset of 15 wavelengths at the left end was chosen so that the wavefront would not fall directly on top of one of the solid lines already plotted. The 2D refraction error = -1.2% relative to the correct value as shown in Fig. 44.

For the case presented there is a difference in SV of 100 m/sec across the boundary. Referring to the SV field for POM Day 08, Figs. 1 and 2, the total range of SV was only 14 m/sec. Figure 45 shows the results for SV differences

Figure 45



of 14 m/sec across the boundary. In this figure the wavefront spacing is for every 100th wave to aid in visualization. The 2D Refraction Error is –0.2%. For many purposes, this would be considered an insignificant error. Here the small angular refraction of about 0.3° is difficult to discern visually. With a refraction error of only –0.2% from the 2D-mode approach, one must conclude that results from the 2D-mode are substantially correct for weakly refracting environments.

8.1.5 The 3D-Mode

In the above paragraphs, the 3D-mode terminology was applied to the Snell's Law solution to the problem. A solution method numerically equivalent to Snell's Law (the proof is obvious and is omitted) is to determine true location of the wavefronts by a method quite similar to that described for the 2D-mode.

To locate points on a wave front by the 3D-mode method, determine the distance along any two lines normal to the departing wavefronts from the SV boundary to the particular wavefront. As in the 2D-mode, the wavefront location is a straight line extending through these two points. Again, the refraction angle can be determined easily. The distances are determined by multiplying the wavelength by the number of wavelength offsets from the SV boundary. Does this procedure sound familiar? The difference is that the normals to the departing wavefronts are in a different plane from the normals to the arriving wavefronts.

This simple case illustrates the order of magnitude of the horizontal refraction error occurring for 2D-mode vs. 3D-mode of putational models. It is fortunate that such close agreement occurs in media having weak horizontal refraction as this greatly extends the usefulness of 2D-mode acoustic models.

8.2 Horizontal Refraction by a 2D-Mode Raytrace

Although 3D raytrace codes are possible, most raytrace codes perform calculations in a 2-dimensional plane. In this section the use of the terminology 2D-mode and 3D-mode has a somewhat different connotation. Out the outset, let us agree that the environment ultimately controls the true ray path.

8.2.1 The 2D-Mode Tacit Assumption

A vertical 2D ray trace calculation is confined to the plane of the radial direction chosen. Such calculations invoke the assumption that horizontal curvature of the ray is zero. That this, in general, is not true provides motivation to refer to the approach as a 2D-mode calculation. By contrast, in a fully developed, 3D ray trace model, horizontal curvature of the ray is possible hence a 3D-mode calculation.

8.2.2 An Idealized Case Where the Exact Solution is Available

Imagine an idealized, though unrealistic, 3-dimensional, SV field that is vertically constant but with a constant horizontal cross-range gradient, say in the Y direction. The SV field presented earlier in Fig. 6 having zero gradients in the X and Z directions and a Y direction gradient of -0.01 sec^{-1} will suffice. Clearly sound rays emanating horizontally would be refracted in the horizontal plane only.

8.2.3 The 2D vs. 3D-Mode Analogy

Since there is no vertical refraction and there are no vertical reflective boundaries, given the source location and initial ray direction, the path can be calculated exactly, by analogy, a 3D-mode solution. On the other hand, if the ray once started is confined to the vertical plane of its origin, a different result will be determined, by analogy the 2D-mode solution. In this section, the 2D-mode calculation is made for the straight-line computational tracks shown in Fig. 37. In 3D-mode, the starting direction of the rays is adjusted so that the true curved paths start and end at points coinciding with the straight line paths.

In general, differences would be expected between arrival phase independently estimated from 2D-mode and 3D modes. Recall that observationally except for very restrictive experimental setups, what may be observed is the arrival phase modulo 360°. In the case of multiple hydrophones, with special care, the additional information of phase relative to a reference hydrophone, again modulo 360° may be determined. With multiple hydrophones if noise is not too great, direction of arrival of the signal may be determined via beamforming. In the absence of horizontal refraction, the signal from a source as shown in Fig. 37, would be the broadside beam of the beamformer. Alternatively, if the source appeared to arrive from some other direction, refraction would of necessity appear to have occurred. Recall that beam direction determined from outputs of the FOR3D Model for this same SV field of Fig. 6 gave the same results for the 2D and 3D-modes, Figs. 40 and 43.

8.2.4 Development of the 2D and 3D Mode Raytrace Equations

8.2.4.1 3D-MODE

The principle of circularity of rays in linear gradients is commonly employed to develop ray paths. Urick [8] describes the approach for a 2D, vertical slice, ray-trace in the ocean. The following presentation parallels that development, except that notation has been changed to represent the problem in a horizontal orientation. The diagram shown in Fig. 46 is similar to the one by Urick.

The line at the left of the figure is the SV, C, having a linear gradient defined by the equation:

$$C = C_0 + gY \qquad \textit{8.2.4.1-1}$$

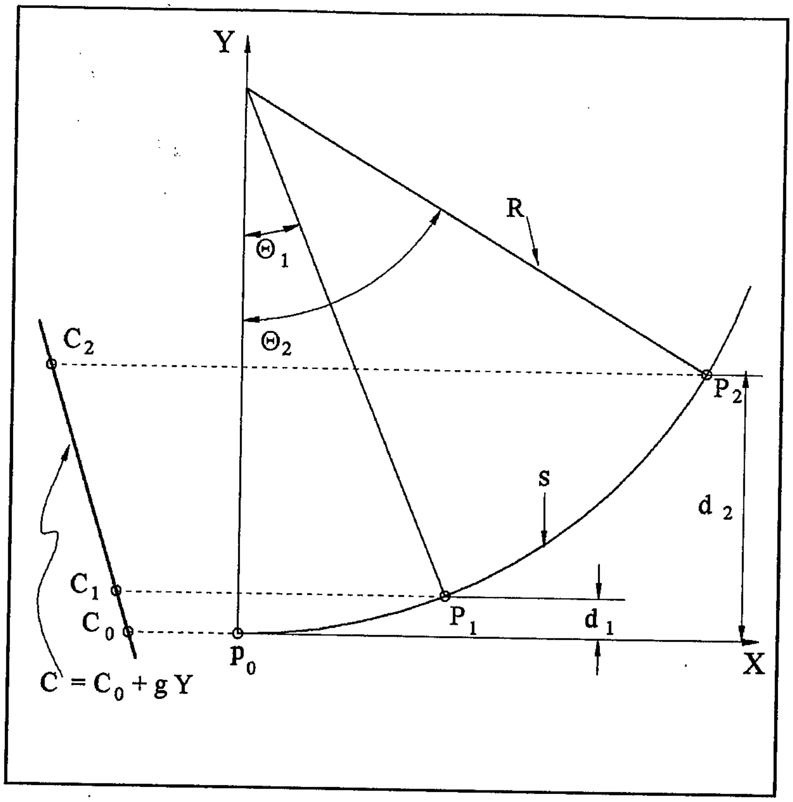
Where:

g = the gradient, sec^{-1}
C_0 = the SV at point P_0
Y = horizontal distance.

Under the constant gradient condition, a ray starting horizontally in direction X from point P_o will travel on the arc shown.

Figure 46

RAYTRACE DIAGRAM



Applying Snell's Law's, the radius of the arc, *R*, may be determined by,

$$Cos\theta_1 = C_1/C_0 \qquad 8.2.4.1\text{-}2$$

$$Cos\theta_2 = C_2/C_0 \qquad 8.2.4.1\text{-}3$$

$$d_2 - d_1 = (C_2 - C_1)/g. \qquad 8.2.4.1\text{-}4$$

From the geometry it follows that:

$$R = -C_0/g \qquad 8.2.4.1\text{-}5$$

In the development above, the ray was assumed to originate horizontally at point P_0. If several rays are to start from a common point but in different directions, a more convenient reference is needed, for example point P_1. Defining P_1 as the new reference point, with reference velocity, $V_0 = C_1$, the new reference velocity, V_0, may be related directly to the ray angle at this new reference point. Consider,

$$V_0 = C_1 = C_0 Cos\theta_1 \qquad 8.2.4.1\text{-}6$$

$$R = -C_0/g = -V_0/gCos\theta_1 \qquad 8.2.4.1\text{-}7$$

In this form, the arc radius is seen to be a function of the direction of the ray at the new reference point P_1. In general, since $C = C_0 Cos\theta$ along the arc path,

$$dS = Rd\theta \qquad 8.2.4.1\text{-}8$$

$$C(\theta) = (V_0/Cos\theta_1)Cos\theta. \qquad 8.2.4.1\text{-}9$$

The transit time, t, along the arc, S, from S_1 at point P_1 to S_2 at point P_2, is defined by the path integral, which after substitution becomes an integral in θ only,

$$t = \int_{S_1}^{S_2} \frac{dS}{C(s)} = \int_{\theta_2}^{\theta_2} \frac{Rd\theta}{\left(V_0 \big/ Cos\theta_1\right) Cos\theta} = -\frac{1}{g}\int_{\theta_1}^{\theta_2} \frac{d\theta}{Cos\theta}. \qquad 8.2.4.1\text{-}10$$

The solution for this integral is tabulated by Gradshteyn [9] as:

$$\int \frac{d\theta}{Cos\theta} = \ln\left[\tan\left(\frac{\pi}{4} + \frac{\theta}{2}\right)\right] + c. \qquad 8.2.4.1\text{-}11$$

In Fig. 46, angles, θ_i, are shown as the angle of the radius vector reference the Y-axis. Since the radius vector is normal to the arc, the angle of the ray at point P_i, reference the horizontal axis, is also θ_i. After integration the final equation for transit time, t, along arc S from point S_1 to S_2 is,

$$t = -\frac{1}{g}\left\{\ln\left[\tan\left(\frac{\pi}{4} + \frac{\theta_2}{2}\right)\right] - \ln\left[\tan\left(\frac{\pi}{4} + \frac{\theta_1}{2}\right)\right]\right\} \qquad 8.2.4.1\text{-}12$$

Since vertical rays are not possible, the solution is limited to the range,

$$-\pi/2 < \theta < \pi/2.$$

In order to utilize Eq. 8.2.4.1-12, θ_1 and θ_2 must be known. The technique used herein involves a relatively simple geometric iteration to determine the angles corresponding to points at the start and ends of the straight line tracks shown in Fig. 37. So that the results can be compared directly to the FOR3D Model, all 2751 tracks will be used even though not all are shown in the figure.

8.2.4.2 2D-MODE

Transit time along any of the tracks shown in Fig. 37 is the path integral along the track. Given track length, L, starting angle, θ, and reference SV, V_0, at the beginning of the track, the time is found by:

$$t = \int_{S_1}^{S_2} \frac{dS}{C(S)} = \int_0^L \frac{dS}{V_0 + gSin(\theta)S} \qquad 8.2.4.2\text{-}1$$

After integration and rearrangement, the transit time is:

$$t = \frac{1}{gSin\theta}\left[\ln(V_0 + gSin(\theta)L) - \ln(V_0)\right] \qquad 8.2.4.2\text{-}2$$

8.2.5 Comparison of Results Predicted by the 2D and 3D-raytace Modes

In the studies of the FOR3D Model, phase data determined at the end of the tracks shown in Fig. 37 were presented in Figs. 39 and 41 where it was shown that the two modes gave virtually identical results for the heart cut radials. Results determined for the same SV field as used for those figures (a field which satisfies the limitations required for the equations just developed), are shown in Figs. 47, 48 and 49 for those same tracks.

In Fig. 47, the transit time from the source to the ends of the tracks is plotted for the assumed straight ray paths and for the true circular arcs. Note that the arc paths have lower transit times as would be expected. From Fig. 48, a plot of the transit time differences, note that the difference is virtually constant.

For simplicity, sound arriving at the end of a track may be thought of a hydrophone signal. Since all wavefronts emanate from a common point, the relative phase at each hydrophone may be determined from,

$$\psi_i = 360[(t_i - t_o)/f]. \qquad 8.2.5\text{-}1$$

To make comparison with the FOR3D Model results simpler, the relative phase, modulo 360, is determined for both the arc and straight paths. These results are shown in Fig. 49. Note that the phase of the arc paths are offset +0.05° so the curves will not be superimposed. It is clear that the relative phases predicted are virtually identical.

An estimate of the error can be determined. In Fig. 48, the time difference varies from about 0.0041 to 0.0042 sec., a total variation of about 0.0001 sec. At 150 Hz, this represents a phase error of,

Phase error= 360(0.0001)/(1/150)=5.4°,

Figure 47

COMPARISON OF TRAVEL TIME ESTIMATES

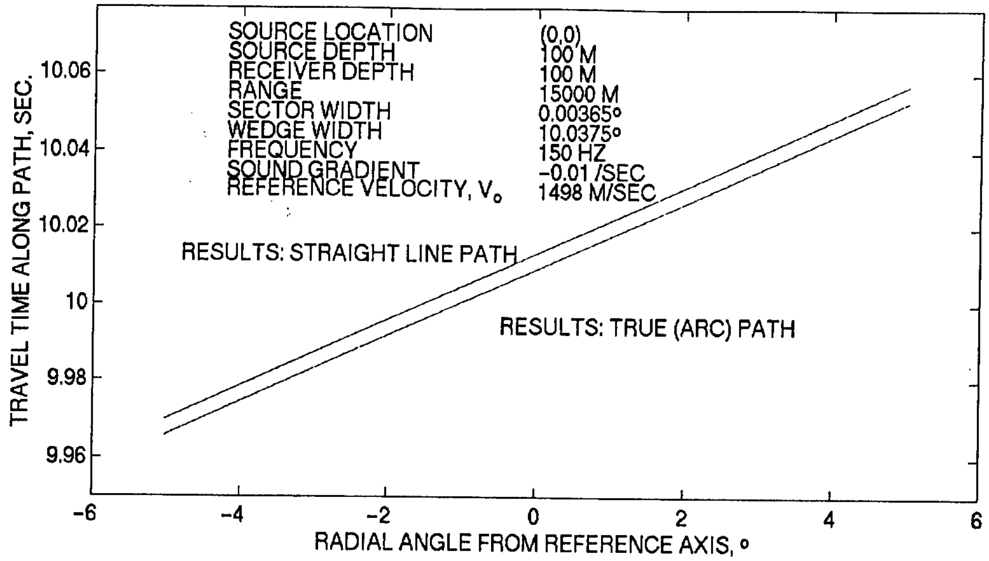


Figure 48

TIME DIFFERENCE, STRAIGHT PATH LESS ARC PATH

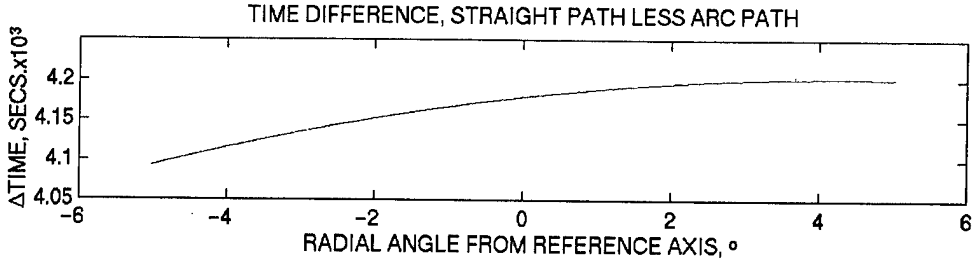
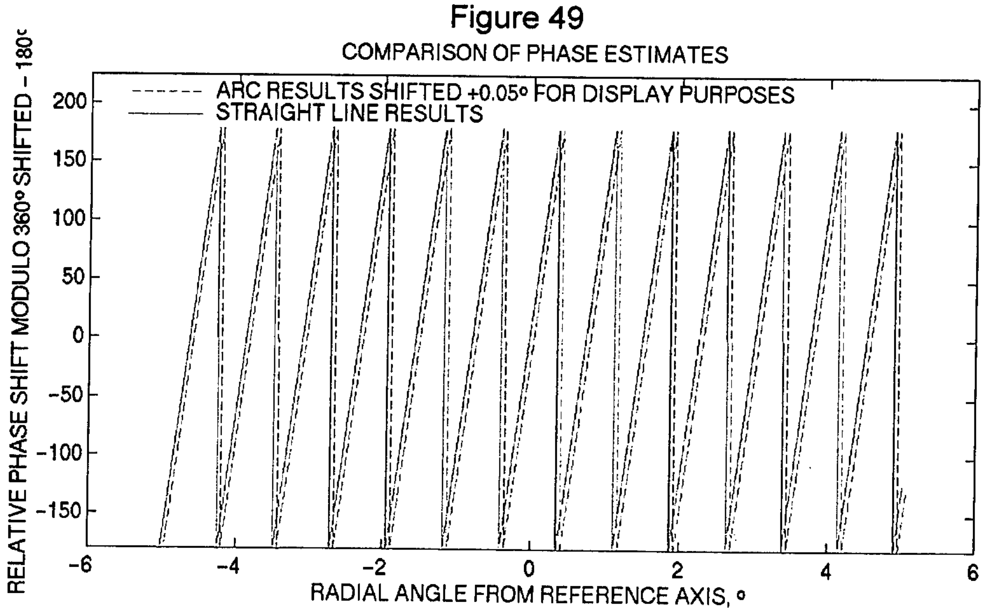


Figure 49

COMPARISON OF PHASE ESTIMATES



about 1.5% of a single cycle over the entire wave front. As a consequence of this close agreement, the 2D-mode is estimating the horizontal refraction. Thus it should be clear that it is the relative phase horizontally across the wavefront rather than the absolute phase (phase relative to the signal origin) that is important.

9. CONCLUSIONS AND RESULTS

This study has shown that the FOR3D Model 3D-mode offers no great improvement over its companion 2D-mode for situations having flat bathymetry in regions having sound velocity (SV) gradients < 0.01 sec^{-1}. As this gradient is much greater than anticipated for the coastal regions of interest, as described earlier, the 2D-mode should be adequate for such studies.

A single trial, made for a cross range sloped bathymetry with gradient of approximately 100 m/15 km, confirmed that the 2D-mode result was adequate for gently sloping conditions in shallow water environments as well.

Two analogous cases have been presented, an idealized horizontal refraction case with a strong SV boundary and a case having a constant horizontal SV gradient. Although the approach to these cases is different, they were developed rigorously and are computationally independent of the FOR3D Model. Each has shown that the horizontal refraction predicted by the 2D-mode is virtually identical to that of the 3D-mode. That this result is true may be surprising to some in view of the fact that the FOR3D Model solution, in 2D-mode, in effect, ignores all azimuthal environmental derivatives.

Upon first consideration, transit time differences as great as noticed between the two modes, might be expected to have consequences easily observable on the dB TL plots. That this did not occur can be explained. In 2D-mode vs. 3D-mode, it is only the relative phase of wavefronts that is critical. If the relative phase relationships are preserved, the resulting interference patterns would be identical. That this has occurred is evident in many of the comparative figures presented.

Software and hardware have been configured such that in 2D-mode, quite large problems can be solved efficiently. Higher frequency solutions, up to 1000 Hz should be possible on the DEC Alpha computer even though this has not been demonstrated directly during this study. An interface to the Princeton Ocean Model has been implemented that is well documented and convenient to use. By extension, other models producing SV fields in rectangular coordinates would also be simple to use.

Software for graphical display, beamforming, and convenient model data entry were produced, much written in Matlab, a very portable language.

10. BIBLIOGRAPHY

1. Lee, Ding, The state-of-the-art parabolic equation approximation as applied to underwater acoustic propagation with discussions on intensive computations, Naval Underwater Systems Center, New London Laboratory, Report No. TD 7247, October 1, 1984.

2. Schultz,M. H. and Saied, Faisal, Solving the wave equation on hypercube architecture, pp. 267-276, Computational Acoustics, Ocean Acoustic Models and Super computing, Vol 1, Lee, D., Cakmak, A. and Vichnevetsky, R. (Editors), Elsevier Science Pub., (1990).

3. Lee, Ding and Pierce, Allan D., Parabolic equation developments in recent decade Jou. Of Comp. Acoustics, Vol. 3, No. 2, pp 95-173, (1995).

4. Lee, Ding and Botseas, George, Examination of three dimensional effects using a propagation model with azimuth-coupling capability (FOR3D) ,J. Acoust. Soc. Am. Vol. 91 No. 6, pp 3192-3202, June (1992).

5. Lee, Ding, Saad, Youcef, and Schultz, Martin H., An efficient method for solving the three-dimensional wide angle wave equation, Report No. YALEU/DCS/RR-463, Yale Univ., Dept of Computer Science, October (1986).

6. Botseas, George, Lee, D., and King, D., FOR3D: A computer Model for solving the LSS three-dimensional wide angle wave equation, Report No. 7943, Naval Underwater Systems Center, August 14, 1987.

7. Lee, Ding, Three-dimensional effects: Interface between the Harvard Open Ocean Model and a three-dimensional acoustic Model, Chapter 5, In Oceanography and Acoustics Prediction and Propagation Models, Edited by Robinson, A. R. and Lee, Ding, American Institute of Physics, 1994.

8. Urick, R. J., Principles of underwater sound, McGraw-Hill Book Co., 1975.

9. Gradshteyn, I. S. and Ryzhik, I. M., Tables of integrals series and products, Equation 14, p 54, Academic Press, (1965).

APPENDIX

A. FOR3D MODEL PROGRAMS

A.1 The FOR3D Model Subroutine List, in Program Order
A.2 Comments About the Program Modules

B. DATA ENTRY AND PREFORMATTER MODULES

B.1 Input Data Files
B.2 Preformatter and Self Documenter

C. PRINCETON OCEAN MODEL FILE FORMAT

D. SPECIAL NOTES REGARDING THE FOR3D MODEL

D.1 FOR3D Model Program Changes

D.2 Changes to the DEC Alpha Unix Kernel Setup

D.3 Compiler Switches Used

D.4 Copies of the Program Modules with Prefix ocean_

E. KEY PROGRAMS USED IN THE FOR3D MODEL PROJECT

E.1 Preformatter Programs

E.2 Plotting FOR3D Model Results

E.3 Beaforming of FOR3D Model Results

APPENDIX A

FOR3D MODEL PROGRAMS

A.1 The FOR3D Model Subroutine List, in Program Order

A.1.1 The Main Program Modules

for3d_main.f
for3d_amifd3.f
for3d_bcon3d.f
for3d_bmifd3.f
for3d_hnkl.f
for3d_indx3d.f
for3d_port2d.f
for3d_port3d.f
for3d_printp.f
for3d_rhs.f
for3d_scon3d.f
for3d_sfld3d.f
for3d_stbd2d.f
for3d_stbd3d.f
for3d_svp3d.f
for3d_trid3d.f
for3d_twostep.f

A.1.2 Those Modules that are Specific to the Harvard Format

harv_ubcon3d.f
ocean_ubottom.f
harv_uexact.f
harv_uport3d.f
harv_uscon3d.f
harv_usfld3d.f
harv_ustbd3d.f
ocean_usvp3d.f
harv_build.f

A.2 Comments About the Program Modules

A.2.1 Compiling the Program

The file, **cat_ocean.com**, is a script file that concatenates all of the above modules into a single program file, **for3d_all_ocean.f.**

The file, **fort77.com**, is a script file for the f77 compiler, invoking the compiler switches that have been used successfully

A.2.2 Removal of Repetitive Reads of the SV and Bathymetry Files

The program modules **ocean_ubottom.f and ocean_usvp3d.f** have been modified from the original source code so that the environmental data files are read only once. This technique speeded up program execution many fold.

A.2.3 Other Program Changes

The programs were reformatted with indentation to make them easier to understand, new comments added for clarity, some I/O conflicts resolved and unused code removed. No changes were made that affected the original computational algorithms. In fact, the existing program set will still read data for the original test case supplied at the download site referred to there as the Harvard format.

APPENDIX B

DATA ENTRY AND PREFORMATTER MODULES

B.1 Input Data Files

The FOR3D Model program requires the two input files:

1. harvard.in
2. harvard.spd

The format and organization of the data in these file is arcane. Though fairly well documented in the source code and reference works cited, in this form it difficult to manually craft input files that will accurately run the intended cases.

B.2 Preformatter and Self Documenter

To facilitate data entry and develop an interface to the data files generated by the Princeton Ocean Model, the program, **harv_build.f**, was developed that prepares the files mentioned in Appendix B.1. This approach was taken, to minimize encroachment of coding errors into the original model. This program also requires two input files:

1. harvard.cfg
2. sound-3D.????   Where: ???? is unique to the SV file used.

The file, **harvard.cfg**, is virtually identical to Table 3 presented in the body of the report. Examination of this file demonstrates that the run configuration is self-documented if a copy of the file is archived.

Generation of the exact format of the configuration file is quite simple. If the file, harvard.cfg, does not exist, harv_build.f creates a sample file in the correct format, exits and informs the user that it has been created. The file can then be edited with any ASCII text editor. Care must be taken to not alter the number of lines in the file, however, the exact column location (within limits) of the input parameters is not critical.

The second file, the POM sound velocity field, contains several additonal values relative to the file size, grid spacing, etc.

If both of the files exist, harv_build.f, reads them, prompts the user for several options that it provides, and creates the aforementioned files required by the FOR3D Model. The main options pertain to the location of the computational wedge (over-rides the configuration file) and the type of bathymetry desired.

APPENDIX C

PRINCETON OCEAN MODEL FILE FORMAT

The simplest way to understand the POM file format is by reference to a minimally simple program that will read the file as follows:

```
C Read-POM.f   Read in a SOUND_3D.DAT File
C***********************************************************
C* Note: grid points that are to be regarded as land
C* locations clearly do not have sound velocity data. At
C* those locations, a flag value may be entered (SPVAL) to
C* designate that water is not present there.  All of the
C* simulations used in the study of the FOR3D Model, at index
C* J=IY, the flag value was entered there, indicating that
C* this corresponds to the coast line.  Special handling of
C* data at those locations is required.
C***********************************************************
      DIMENSION SOUND(IX,IY,IY)
C***********************************************************

      OPEN(99, FILE='SOUND_3D.DAT',
     +       STATUS='UNKNOWN',FORM='FORMATTED')

      READ(99,101) IX,IY,IZ
101   FORMAT(4I6)
C        Z(1) is at the surface.  Z(IZ) is at the bottom.
C        For flat bottom assumption:  H=DZ*FLOAT(IZ)

      READ(99,104) TDAY,SPVAL,DX,DY,DZ
104   FORMAT(5(E12.6))
C        TDAY=time in days for the simulation
C        SPVAL=land points (undefined value)
C        DX, DY, DZ (increment size, m)

      READ(99,105) (((SOUND(I,J,K),I=1,IX),J=1,IY),K=1,IZ)
105   FORMAT(12(E12.6))

      CLOSE(99)
```

APPENDIX D

SPECIAL NOTES REGARDING THE FOR3D MODEL

D.1 FOR3D Model Program Changes

Appendix A, Section A.1, contains a complete list of all program modules comprising the FOR3D Model.

Note that the first group of programs has the prefix for3d_. Note that in the second group of programs most have the prefix harv_. Those modules are virtually identical to the original downloaded programs but have been modified slightly. Cosmetic changes, indentation, some I/O inconsistencies fixed, changes of do loop structures to use the enddo construct, and addition of clarifying comments have been made. Changes were also required to array sizes and some of the logic to allow reading in the sound field and bathymetry file only one time.

Note that in the second group, two of the modules have the prefix ocean_. These two were modified sufficiently to enable reading of the sound field and bathymetry field only once rather than over and over as was required by the as received model.

D.2 Changes to the DEC Alpha Unix Kernel Setup

In order to allow large Fortran programs to load successfully, several changes were required to the default operating kernel as setup by the DEC Alpha Unix operating system installation disk. In order to make these changes, login as a super user, run dxkerneltuner and make the following changes:

Under **proc** setup the following as shown below:

maxusers	12
max-per-proc-address-space	8589934592
per-proc-address-space	8589934592
max-per-proc-data-size	8589934592
per-proc-data-size	8589934592

Under **vm** setup the following as shown below:

vm-maxvas	9992929280
vm-mapentries	200
vm-maxwire	16777216

The above changes allowed loading and running Fortran programs of 2 Gbytes compiled size although the actual total memory was only 512 Mbytes. Of course to accomplish this load size, sufficient virtual memory must be available on the

hard disk. Prior to these kernel changes, such large programs would not even load on the Dec Alpha, much less run.

It is not known if the changes listed above are optimal or even all required, however, these changes were successfully used as described.

D.3 Compiler Switches Used

The Fortran compiler used was the standard Dec Fortran 77 complier compatible with the DEC Unix Operation System on the Dec Alpha computer. The compiler options used are invoked as follows:

f77 –extend_source –check underflow –check overflow –g –C –f –o $1 $1.f

D.4 Copies of the Program Modules with Prefix ocean_

Program file 'ocean_ubottom.f'

```
C     File   ocean_ubottom.f                                       ***********
C                                                                  * UBOTTOM *
C                                                                  ***********
C     Last revision 1/14/99 A. E. Leybourne
C
      SUBROUTINE UBOTTOM
C   Filename is ocean_ubottom.f, INTENDED TO REPLACE harv_ubottom.f
C*************************************************************************
C   The following definition of the harvard input file format is as I
C   A. E. Leyboure understand it.  There may be some errors in this at
C   the present time. These notes are what I have been able to glean
C   from notes elsewhere and supersluething the for3d.f code.
C
C---------------- Beginning of file formatting notes -------------------
C
C   Varibles that are in common are marked as GLOBAL
C
C------------------------------------------------------------------------
C READ Statement 1
C
C          READ(NHU,103)DLAT0,DLNGO,THTA,NX,NY
C    103   FORMAT(2X,2F7.3,F8.5,2I6)
C            DLATO  Also in USVP3D -- Latitude of domain (ARRAY) center
C            DLNGO  Also in USVP3D -- Longitude of domain (ARRAY) center
C            THTA   Also in USVP3D -- CCW domain rotation in RADIANS,
C                                     CCW angle of array X axis to a
C                                     line parallel to the equator
C            NX     Also in USVP3D -- No steps in X direction
C            NY     Also in USVP3D -- No steps in Y direction
C                   (Not an input, but)  NXY=NX*NY
C
C READ Statement 2
C
C          READ(NHU,104)DX,DY,NLEV
C    104   FORMAT(2(F7.1),I6)
```

```
C               DX      Also in USVP3D -- Increment X direction, kmeters
C               DY      Also in USVP3D -- Increment Y direction, kmeters
C                       (After input these are converted to meters)
C               NLEV    Also in USVP3D -- No of levels in the file
C                       and   NLEV .LE. MXLEX
C  GLOBAL               NSVP   (Not an input, but)  NSVP=NLEV+1
C                       XBASIN=(NX-1)*DX
C                       YBASIN=(NY-1)*DY
C                       ANGLE=THTA+ATAN2(YBASIN,XBASIN)
C                       DCENTER=SQRT((.5*XBASIN)**2+(.5*YBASIN)**2)
C
C READ Statement 3 (The number of RECORDS may vary)
C
C           READ(NHU,105)  (ZLEV(I),I=1,NLEV)
C    105    FORMAT(10F7.1)
C             ZLEV   Also in USVP3D -- Values for each data level
C             Note the sucess of this read depends on each record
C             containing 10 data fields, except the last record which
C             can be only partially filled; since Fortran reuses the
C             FORMAT repetitively until the input list is satisfied.
C             The original sample harvard input data file was not
C             compatible with format statement as coded originally.
C             See sample input file e_ln3_f.spd
C
C READ Statement 4
C
C          READ(NHU,107,END=192)NN,T,ITYP,K,ITMP1,ITMP2
C    107   FORMAT(2X,I5,F10.5,4I5)
C             NN      Also in USVP3D -- No elements at each level
C                     (NN must = NXY
C             T       Also in USVP3D -- Not used in UBOTTOM
C             ITYP    Also in USVP3D -- Not used in UBOTTOM
C             K       Also in USVP3D -- Not used in UBOTTOM
C             ITMP1   Also in USVP3D -- Not used in UBOTTOM
C             ITMP2   Also in USVP3D -- Not used in UBOTTOM
C
C READ Statement 5
C
C          NXY = NX*NY
C          READ(NHU,108)(BOT(I),I=1,NXY)
C    108   FORMAT(8F10.3)
C               BOT   Also in USVP3D -- Bottom depth array
C
C------------------------------------------------------------------------
C    The little routine below illustrates fortran array filling order,
C    in this case  row1, 1 to NX; row2, 1 to NX; etc. to NY rows
C------------------------------------------------------------------------
C      PARAMETER(NX=5,NY=3,NXY=NX*NY)
C
C      DIMENSION A(NX,NY), B(NXY)
C      EQUIVALENCE(A,B)
C      DO J=1,NY
C        DO I=1,NX
C          A(I,J)=100*I + J
C        ENDDO
C      ENDDO
C
```

```
C       PRINT *
C       DO J=1,NY
C         PRINT *,(A(I,J),I=1,NX)
C       ENDDO
C
C       PRINT *
C       DO J=1,NY
C         PRINT *,(B((J-1)*NX+I),I=1,NX)
C       ENDDO
C       END
C
C     The code snitches shown below appear to be accessing the data
C     in the same way,
C
C         I1=(JS-1)*NX+IS
C         I2=I1+1
C          I3=I1+NX
C          I4=I1+NX+1
C          IF(H1.LE.H) THEN
C            DEPTH=BOT(I1)+XOFF*(BOT(I2)-BOT(I1))/DX+
C    +         YOFF*(BOT(I3)-BOT(I1))/DY
C          ELSE
C            DEPTH=BOT(I4)+(DX-XOFF)*(BOT(I3)-BOT(I4))/DX+
C    +         (DY-YOFF)*(BOT(I2)-BOT(I4))/DY
C          ENDIF
C
C     Thus we may assume that the data file order is the same as in
C     the above, i.e.,
C
C             All X values for Y=  DY*(1-1)
C                 X values for Y=  DY*(2-1)
C                                .
C                                .
C                 X values for Y=  DY*(NY-1)
C
C     The establishment of data order is extremely important if the
C     data file generated (in harvard format) for ocean model data is
C     to be interpreted correctly.
C
C     The justification for using READ statements like 3 and 5 is
C     to make the read statement generic in array size as long as it is
C     large enough to hold all of the data.
C
C------------------ Ending of file formatting notes ---------------------
C
C***********************************************************************
C                 BEGINING OF UBOTTOM PROGRAM CODE                     *
C***********************************************************************
      INCLUDE 'for3d.cmn'
C-----------------------------------------------------------------------
C     This common block and PARAMETERS are used by SUBROUTINES
C      USVP3D and UBOTTOM for the HARVARD data entry method only
C-----------------------------------------------------------------------
      PARAMETER (MXLEV=40,MXY=7991)
C     *** MXLEV is maximum number of levels in input data set
C     *** MXY   is maximum number of data points in each level
      COMMON /HARVARD/BOT(MXY),ZLEV(MXLEV),
```

```
     +                    SVP(MXY,MXLEV),SLAT(MXSOL),SLNG(MXSOL),
     +                    DLAT0,DLNG0,SLAT0,SLNG0,DIR,THTA,
     +                    BOTRHO,BOTRHOG,BOTBETA,BOTBETAG,CWCB,CGRAD,SEDZ,
     +                    CONVLAT,CONVLNG2,EPS,DXEPS,
     +                    DX,DY,XBASIN,YBASIN,ANGLE,DCENTER,
     +                   NDAY,NX,NY,NXY,NLEV,NN,IUSVP3D,
     +                   USVP_READIN

      REAL               BOT,ZLEV,
     +                    SVP,SLAT,SLNG,
     +                    DLAT0,DLNG0,SLAT0,SLNG0,DIR,THTA,
     +                    BOTRHO,BOTRHOG,BOTBETA,BOTBETAG,CWCB,CGRAD,SEDZ,
     +                    CONVLAT,CONVLNG2,EPS,DXEPS,
     +                    DX,DY,XBASIN,YBASIN,ANGLE,DCENTER

      INTEGER*4          NDAY,NX,NY,NXY,NLEV,NN,IUSVP3D

      LOGICAL            USVP_READIN
C-----------------------------------------------------------------------
      DIMENSION DZOLD(MXSOL),DZNEW(MXSOL)
C     Logical to hold result of queries
      LOGICAL Q_OPENED,EXIST_FLG
      DATA RAD/6378400/
C-----------------------------------------------------------------------
      PRINT *,'UBOTTOM Called, KSVP = ',KSVP
C   Next lined added to get rid of compiler unused variable flag
      IUSVP3D=IUSVP3D

C     *** input parameters for harvard data from file harvard.in
C              *** U's Already read in earlier ***
      NDAY=U1
C        day of data set : 3 means 3rd day
      SLAT0=U2
C        latitude of starting field
      SLNG0=U3
C        longitude of starting field
      DIR=U4
C        direction, propagation center ray, deg, measured CCW from North
      BOTRHO=U5
C        density in bottom
      BOTRHOG=U6
C        density gradient in bottom
      BOTBETA=U7
C        attenuation in bottom
      BOTBETAG=U8
C        attenuation gradient in bottom
      CWCB=U9
C        sound speed ratio at bottom interface
      CGRAD=U10
C        sound speed gradient in bottom
      SEDZ=U11
C        sediment thickness

      DO J=1,NSOL
        SLAT(J)=U2
        SLNG(J)=U3
        ITRK(J)=0
```

```
        DZOLD(J)=0
      ENDDO
C
C     *** compute meters per degree of latitude at the equator
      CONVLAT=PI*RAD/180.0

C     *** read data in harvard file format
C     *** NHU   is harvard input unit number
C--------------------------------------------------------------------
      INQUIRE(FILE='harvard.spd',EXIST=EXIST_FLG)
      IF(.NOT. EXIST_FLG) THEN
        PRINT *
        PRINT *,'File harvard.spd not found. Copy YOUR sound'
        PRINT *,'   speed file to harvard.spd, then rerun'
        PRINT *
        STOP
      ENDIF

C     If input file is not open, then open it
      INQUIRE(UNIT=NHU,OPENED=Q_OPENED)
      IF(.NOT. Q_OPENED) THEN
        OPEN(UNIT=NHU,FILE='harvard.spd',STATUS='OLD')
      ENDIF

      READ(NHU,103)DLAT0,DLNG0,THTA,NX,NY
103   FORMAT(2X,2F7.3,F8.5,2I6)

      READ(NHU,104)DX,DY,NLEV
104   FORMAT(2(F7.1),I6)

C*    The as received program code number in format below did
C*    not agree with the file format itself, changed to 10
      READ(NHU,105) (ZLEV(I),I=1,NLEV)
105   FORMAT(10F7.1)
C  ------------------------------------------------------------------
      IF(NLEV.GT.MXLEV) THEN
        WRITE(NPU,*)'ERROR. TOO MANY LEVELS. INCREASE MXLEV. RECOMPILE.'
        NSVP=0
        RETURN
      ENDIF

C     *** grid size in meters
      DX=DX*1000.0
      DY=DY*1000.0
C     Compute m per degree longitude at latitude DLAT0
      CONVLNG2=RAD*COS(PI*DLAT0/180.)*PI/180.0
      EPS=ATAN2(DY,DX)
      DXSEPS=DX*SIN(EPS)
      NXY=NX*NY
      XBASIN=(NX-1)*DX
      YBASIN=(NY-1)*DY
C   Compute radian angle CCW from equator to array diagonal
C      (origin to extreme)
      ANGLE=THTA+ATAN2(YBASIN,XBASIN)
      DCENTER=SQRT((.5*XBASIN)**2+(.5*YBASIN)**2)

      READ(NHU,107,END=192)NN,T,ITYP,K,ITMP1,ITMP2
```

```
107    FORMAT(2X,I5,F10.5,4I5)

       IF(NN.NE.NXY)GO TO 190

C      *** read bottom depths
       READ(NHU,108)(BOT(I),I=1,NXY)
108    FORMAT(8F10.3)

C      *** End of bottom depths, sound speed not read in and no other
C          data read in
       REWIND(UNIT=NHU)

C     Now use the data readin and pass results to the calling program
C     through the COMMON statement
       ZMAX=0.0
C      Compute m per degree longitude at latitude SLAT0
       CONVLNG0=RAD*COS(PI*SLAT0/180.)*PI/180.0
       DO J=1,NSOL
       DZOLD(J)=0.0
       ENDDO
       DO R=RA,RMAX,DR
         DO J=1,NSOL
C          Compute track angle in radians CW from equator
C          where DIR is angle in degrees CW from north
C            The original code was
C              STHETA=DIR*PI/180.0-(FLDW/2.0-(J-1)*DTH)*PI/180.0
C              STHETA=90.0*PI/180.0-STHETA
C            Revised code (easier to understand)
               STHETA=(90.0-DIR+FLDW/2.0-(J-1)*DTH)*(PI/180.0)

C          Compute latitude at range R
             SLAT(J)=SLAT0+R*SIN(STHETA)/CONVLAT

C          Compute m/deg longitude at latitude SLAT(J), R of track
             CONVLNGR=RAD*COS(PI*SLAT(J)/180.)*PI/180.0
C          Compute m/deg longitude average for track
             CONVLNG=.5*(CONVLNG0+CONVLNGR)
C          Compute longitude at range R
           SLNG(J)=SLNG0-R*COS(STHETA)/CONVLNG
C          Compute horz distance array origin to current end of track
           A1=((DLNG0+DCENTER*COS(ANGLE)/CONVLNG2)-SLNG(J))*CONVLNG2
C          Compute vert distance array origin to current end of track
           B1=(SLAT(J)-(DLAT0-DCENTER*SIN(ANGLE)/CONVLAT))*CONVLAT
C          Compute distance array center to current end of track
           C1=SQRT(A1**2+B1**2)
C          Compute angle of track to array X axis
           PSI=ATAN2(B1,A1)-THTA
C          Compute NUMBER array X elements from center to end of track
           RIS=(C1*COS(PSI)/(DX))+1
           IS=INT(RIS)
           IF(IS.GT.NX) THEN
             WRITE(NPU,194)
             PRINT 194
             STOP
           ENDIF
C
C          Compute NUMBER array Y elements from center to end of track
```

```
      RJS=(C1*SIN(PSI)/(DY))+1
      JS=INT(RJS)
      IF(JS.GT.NY) THEN
        WRITE(NPU,195)
        PRINT 195
        STOP
      ENDIF
C
      IF(IS.LT.1) THEN
        WRITE(NPU,194)
        PRINT 194
        STOP
      ENDIF
C
      IF(JS.LT.1) THEN
        WRITE(NPU,195)
        PRINT 195
        STOP
      ENDIF
C
      XIP=C1*COS(PSI)
      YJP=C1*SIN(PSI)
      A2=DX*(IS-1)
      B2=DY*(JS-1)
      C2=SQRT(A2**2+B2**2)
      PSI2=ATAN2(B2,A2)
      XIS=C2*COS(PSI2)
      YJS=C2*SIN(PSI2)
      XOFF=(XIP-XIS)
      YOFF=(YJP-YJS)
      ZETA=ATAN2(YOFF,XOFF)
      GAMMA=PI-EPS-ZETA
      H1=SQRT(XOFF*XOFF+YOFF*YOFF)
      H=DXSEPS/SIN(GAMMA)
      I1=(JS-1)*NX+IS
      I2=I1+1
      I3=I1+NX
      I4=I1+NX+1
      IF(H1.LE.H) THEN
        DEPTH=BOT(I1)+XOFF*(BOT(I2)-BOT(I1))/DX+YOFF*
     +          (BOT(I3)-BOT(I1))/DY
      ELSE
        DEPTH=BOT(I4)+(DX-XOFF)*(BOT(I3)-BOT(I4))/DX+(DY-YOFF)*
     +          (BOT(I2)-BOT(I4))/DY
      ENDIF
      IF(ZMAX.LT.DEPTH)ZMAX=DEPTH
      ITRK(J)=ITRK(J)+1
      TRACK(ITRK(J),1,J)=R
      TRACK(ITRK(J),2,J)=DEPTH
      IF(R.GT.RA)THEN
      DZNEW(J)=DEPTH
      IF(ABS(DZOLD(J)-DZNEW(J)).LT..5*ZA/N.AND.ITRK(J).GT.2)THEN
C         *** limit number of track points
          ITRK(J)=ITRK(J)-1
          TRACK(ITRK(J),1,J)=R
          TRACK(ITRK(J),2,J)=DEPTH
        ELSE
```

```fortran
          DZOLD(J)=DZNEW(J)
          ENDIF
        ENDIF
      ENDDO
    ENDDO
C   *** end track.
    DO J=1,NSOL
      TRACK(ITRK(J)+1,1,J)=1.0E+38
      TRACK(ITRK(J)+1,2,J)=TRACK(ITRK(J),2,J)
C     *** initialize track segment arrays.
      R2(J)=TRACK(1,1,J)
      Z2(J)=TRACK(1,2,J)
    ENDDO

C   *** if depth of water plus depth of sediment GT ZA, adjust N.
    IF(ZMAX+SEDZ.GT.ZA) THEN
      DZ=ZA/N
      WRITE(NPU,*)'ZMAX+SEDZ GT ZA. ZA= ',ZA,'. N= ',N,'. DZ= ',DZ,'.'
      WRITE(NPU,*)'ZMAX = ',ZMAX,'.  SEDZ = ',SEDZ,'.'
      RN=(ZMAX+SEDZ)/DZ
      N=INT(RN)
      ZA=N*DZ
      WRITE(NPU,*)'N RESET TO ',N,'.   ZA RESET TO ',ZA,'.'
    ENDIF

    IF(N.GT.MXNZ)THEN
      WRITE(NPU,*)'N TOO LARGE. INCREASE PARAMETER MXNZ. RECOMPILE.'
      STOP
    ENDIF
    RETURN

190 WRITE(NPU,191)
191   FORMAT(1X,'DATA MISMATCH. NX*NY DOES NOT EQUAL NN.')
      NSVP=0
    RETURN

C   Get here in error when reading past EOF
192 WRITE(NPU,193)
193   FORMAT(1X,'READ ERROR. CHECK INPUT SOUND SPEED FILE.')
      NSVP=0
    RETURN

194 FORMAT(1X,'LONGITUDE OUTSIDE LIMITS OF SOUND SPEED DOMAIN.')
195 FORMAT(1X,'LATITUDE OUTSIDE LIMITS OF SOUND SPEED DOMAIN.')

    END
```

Program file 'ocean_usvp3d.f'

```
C     File   ocean_usvp3d.f                                         ***********
C                                                                   * USVP3D  *
C                                                                   ***********
C     Last revision 12/9/98 A. E. Leybourne
C
      SUBROUTINE USVP3D

C   Filename is ocean_usvp3d.f, INTENDED TO REPLACE harv_usvp3d.f
C***********************************************************************
C   The following definition of the harvard input file format is as I
C   A. E. Leyboure understand it.  There may be some errors in this at
C   the present time. These notes are what I have been able to glean
C   from notes elsewhere and supersluething the for3d.f code.
C
C---------------- Beginning of file formatting notes -------------------
C
C   Varibles that are in common are marked as GLOBAL
C
C-----------------------------------------------------------------------
C READ Statement 1
C
C          READ(NHU,103)DLAT0,DLNGO,THTA,NX,NY
C    103   FORMAT(2X,2F7.3,F8.5,2I6)
C           DLATO  Also in UBOTTOM -- Latitude of domain (ARAY) center
C           DLNGO  Also in UBOTTOM -- Longitude of domain (ARRAY) center
C           THTA   Also in UBOTTOM -- CCW domain rotation in RADIANS,
C                                     CCW angle of array X axis to a
C                                     line parallel to the equator
C           NX     Also in UBOTTOM -- No steps in X direction
C           NY     Also in UBOTTOM -- No steps in Y direction
C                  (Not an input, but)  NXY=NX*NY
C
C READ Statement 2
C
C          READ(NHU,104)DX,DY,NLEV
C    104   FORMAT(2(F7.1),I6)
C             DX      Also in UBOTTOM -- Increment X direction, kmeters
C             DY      Also in UBOTTOM -- Increment Y direction, kmeters
C                     (After input these are converted to meters)
C             NLEV    Also in UBOTTOM -- No of levels in the file
C                     and   NLEV .LE. MXLEX
C  GLOBAL             NSVP   (Not an input, but)  NSVP=NLEV+1
C                     XBASIN=(NX-1)*DX
C                     YBASIN=(NY-1)*DY
C                     ANGLE=THTA+ATAN2(YBASIN,XBASIN)
C                     DCENTER=SQRT((.5*XBASIN)**2+(.5*YBASIN)**2)
C
C READ Statement 3
C
C          READ(NHU,105) (ZLEV(I),I=1,NLEV)
C    105   FORMAT(10F7.1)
C            ZLEV   Also in UBOTTOM -- Values for each data level
C            Note the sucess of this read depends on each record
C            containing 10 data fields, except the last record which
```

```
C             can be only partially filled; since Fortran reuses the
C             FORMAT repetitively until the input list is satisfied.
C             The original sample harvard input data file was not
C             compatible with format statement as coded originally.
C             See sample input file e_ln3_f.spd
C
C READ Statement 4
C
C          NXY = NX*NY
C          READ(NHU,107,END=192)NN,T,ITYP,K,ITMP1,ITMP2
C    107   FORMAT(2X,I5,F10.5,4I5)
C             NN      Also in UBOTTOM -- No elements at each level
C                     (NN must = NXY
C             T       Also in UBOTTOM -- Not used in USVP3D
C             ITYP    Also in UBOTTOM -- Not used in USVP3D
C             K       Also in UBOTTOM -- Not used in USVP3D
C             ITMP1   Also in UBOTTOM -- Not used in USVP3D
C             ITMP2   Also in UBOTTOM -- Not used in USVP3D
C
C READ Statement 5
C
C          READ(NHU,108)(BOT(I),I=1,NXY)
C    108   FORMAT(8F10.3)
C              BOT    Also in UBOTTOM -- Bottom depth array
C
C----------------------------------------------------------------------
C    The little routine below illustrates fortran array filling order,
C    in this case  row1, 1 to NX; row2, 1 to NX; etc. to NY rows
C----------------------------------------------------------------------
C      PARAMETER(NX=5,NY=3,NXY=NX*NY)
C
C      DIMENSION A(NX,NY), B(NXY)
C      EQUIVALENCE(A,B)
C      DO J=1,NY
C        DO I=1,NX
C          A(I,J)=100*I + J
C        ENDDO
C      ENDDO
C
C      PRINT *
C      DO J=1,NY
C        PRINT *,(A(I,J),I=1,NX)
C      ENDDO
C
C      PRINT *
C      DO J=1,NY
C        PRINT *,(B((J-1)*NX+I),I=1,NX)
C      ENDDO
C      END
C
C     The code snitches shown below appear to be accessing the data
C     in the same way,
C
C         I1=(JS-1)*NX+IS
C         I2=I1+1
C          I3=I1+NX
C          I4=I1+NX+1
```

```
C          IF(H1.LE.H) THEN
C            DEPTH=BOT(I1)+XOFF*(BOT(I2)-BOT(I1))/DX
C     +        +YOFF*(BOT(I3)-BOT(I1))/DY
C          ELSE
C            DEPTH=BOT(I4)+(DX-XOFF)*(BOT(I3)-BOT(I4))/DX+
C     +        (DY-YOFF)*(BOT(I2)-BOT(I4))/DY
C          ENDIF
C
C     Thus we may assume that the data file order is the same as in
C     the above, i.e.,
C
C              All X values for Y=  DY*(1-1)
C                  X values for Y=  DY*(2-1)
C                                 .
C                                 .
C                  X values for Y=  DY*(NY-1)
C
C     The establishment of data order is extremely important if the
C     data file generated (in harvard format) for ocean model data is
C     to be interpreted correctly.
C
C     The justification for using READ statements like 3 and 5 is
C     to make the read statement generic in array size as long as it is
C     large enough to hold all of the data.
C
C------------------ Ending of file formatting notes --------------------
C
C**********************************************************************
C                 BEGINING OF USVP3D PROGRAM CODE                     *
C**********************************************************************
      INCLUDE 'for3d.cmn'
C----------------------------------------------------------------------
C     This common block and PARAMETERS are used by SUBROUTINES
C      USVP3D and UBOTTOM for the HARVARD data entry method only
C----------------------------------------------------------------------
      PARAMETER (MXLEV=40,MXY=7991)
C     *** MXLEV is maximum number of levels in input data set
C     *** MXY   is maximum number of data points in each level
      COMMON /HARVARD/BOT(MXY),ZLEV(MXLEV),
     +                  SVP(MXY,MXLEV),SLAT(MXSOL),SLNG(MXSOL),
     +                  DLAT0,DLNG0,SLAT0,SLNG0,DIR,THTA,
     +                  BOTRHO,BOTRHOG,BOTBETA,BOTBETAG,CWCB,CGRAD,SEDZ,
     +                  CONVLAT,CONVLNG2,EPS,DXEPS,
     +                  DX,DY,XBASIN,YBASIN,ANGLE,DCENTER,
     +                 NDAY,NX,NY,NXY,NLEV,NN,IUSVP3D,
     +                 USVP_READIN

      REAL             BOT,ZLEV,
     +                  SVP,SLAT,SLNG,
     +                  DLAT0,DLNG0,SLAT0,SLNG0,DIR,THTA,
     +                  BOTRHO,BOTRHOG,BOTBETA,BOTBETAG,CWCB,CGRAD,SEDZ,
     +                  CONVLAT,CONVLNG2,EPS,DXEPS,
     +                  DX,DY,XBASIN,YBASIN,ANGLE,DCENTER

      INTEGER*4        NDAY,NX,NY,NXY,NLEV,NN,IUSVP3D

      LOGICAL          USVP_READIN
```

```
C-----------------------------------------------------------------------
C     Logicals to hold result of queries
      LOGICAL Q_OPENED,EXIST_FLG
      DATA RAD/6378400/
C***********************************************************************
*      PRINT *,'USVP3D Called, KSVP = ',KSVP
      IUSVP3D=IUSVP3D+1

      IF(KSVP .NE. 1) THEN
        NSVP=0
        RETURN
      ENDIF
C     *** input parameters for harvard data from file harvard.in
C               *** U's  already read in earlier ***
      NDAY=U1
C        day of data set : 3 means 3rd day
      SLAT0=U2
C        latitude of starting field
      SLNG0=U3
C        longitude of starting field
      DIR=U4
C        direction of propagation of center ray
      BOTRHO=U5
C        density in bottom
      BOTRHOG=U6
C        density gradient in bottom
      BOTBETA=U7
C        attenuation in bottom
      BOTBETAG=U8
C        attenuation gradient in bottom
      CWCB=U9
C       sound speed ratio at bottom interface
      CGRAD=U10
C        sound speed gradient in bottom
      SEDZ=U11
C        sediment thickness

      DO J=1,NSOL
        SLAT(J)=U2
        SLNG(J)=U3
      ENDDO

C     *** Compute meters per degree of latitude
      CONVLAT=PI*RAD/180.0
C
      IF(.NOT. USVP_READIN) THEN
C       *** Read data in harvard file format
C       *** NHU   is harvard input unit number
C       ------------------------------------------------------
        INQUIRE(FILE='harvard.spd',EXIST=EXIST_FLG)
        IF(.NOT. EXIST_FLG) THEN
          PRINT *
          PRINT *,'File harvard.spd not found. Copy YOUR sound'
          PRINT *,'   speed file to harvard.spd, then rerun'
          PRINT *
          STOP
        ENDIF
```

```
C        *** If input file is not open, then open it
         INQUIRE(UNIT=NHU,OPENED=Q_OPENED)
         IF(.NOT. Q_OPENED) THEN
           OPEN(UNIT=NHU,FILE='harvard.spd',STATUS='OLD')
         ENDIF

         READ(NHU,103)DLAT0,DLNG0,THTA,NX,NY
103      FORMAT(2X,2F7.3,F8.5,2I6)

         READ(NHU,104)DX,DY,NLEV
104      FORMAT(2(F7.1),I6)

C*       The as received program code number in format below did
C*       not agree with the file format itself, changed to 10
         READ(NHU,105) (ZLEV(I),I=1,NLEV)
105      FORMAT(10F7.1)
C*       -----------------------------------------------------------

         IF(NLEV.GT.MXLEV) THEN
           WRITE(NPU,*)'ERROR. TOO MANY LEVELS. INCREASE MXLEV.',
     +                 ' RECOMPILE.'
           NSVP=0
           RETURN
         ENDIF
C
C        *** GRID SIZE IN METERS
         DX=DX*1000.0
         DY=DY*1000.0
         CONVLNG2=RAD*COS(PI*DLAT0/180.)*PI/180.0
         EPS=ATAN2(DY,DX)
         DXSEPS=DX*SIN(EPS)
         NXY=NX*NY
         XBASIN=(NX-1)*DX
         YBASIN=(NY-1)*DY
C    Compute radian angle CCW from equator to array diagonal
C        (origin to extreme)
         ANGLE=THTA+ATAN2(YBASIN,XBASIN)
         DCENTER=SQRT((.5*XBASIN)**2+(.5*YBASIN)**2)

         READ(NHU,107,END=192)NN,T,ITYP,K,ITMP1,ITMP2
107      FORMAT(2X,I5,F10.5,4I5)

         IF(NN.NE.NXY)GO TO 190

C        *** Read bottom depths
         READ(NHU,108)(BOT(I),I=1,NXY)
108      FORMAT(8F10.3)

C        *** Read NLEV levels of speed data for day NDAY
C      Outside loop disabled, there will never be more that one day
C      in the ocean model data file, AEL
C         DO IDAY=1,NDAY
          DO IDAY=1,1
           DO L=1,NLEV
             READ(NHU,107,END=192)NN,T,ITYP,K,ITMP1,ITMP2
             IF(NN.NE.NXY)GO TO 190
```

```
            READ(NHU,110)(SVP(I,L),I=1,NXY)
110         FORMAT(5(E12.6))

          ENDDO
        ENDDO

C       *** end of sound speed input, next line added AEL
        REWIND(UNIT=NHU)
      ENDIF

C   Set flag so that data will not have to be read in again
      USVP_READIN=.TRUE.

      RAOLD=RA
C   Compute m per degree longitude at latitude SLAT0, start of track
      CONVLNG0=RAD*COS(PI*SLAT0/180.)*PI/180.0
      DO J=1,NSOL
C         Compute track angle in radians CW from equator
C         where DIR is angle in degrees CW from north
C           The original code was
C             STHETA=DIR*PI/180.0-(FLDW/2.0-(J-1)*DTH)*PI/180.0
C             STHETA=90.0*PI/180.0-STHETA
C           Revised code (easier to understand)
              STHETA=(90.0-DIR+FLDW/2.0-(J-1)*DTH)*(PI/180.0)

C         Compute latitude at range RA
            SLAT(J)=SLAT0+RA*SIN(STHETA)/CONVLAT

C         Compute m/deg longitude at latitude SLAT(J), RA of track
            CONVLNGR=RAD*COS(PI*SLAT(J)/180.)*PI/180.0
C         Compute m/deg longitude average for track
            CONVLNG=.5*(CONVLNG0+CONVLNGR)
C         Compute longitude at range RA
          SLNG(J)=SLNG0-RA*COS(STHETA)/CONVLNG


        CONVLNG=.5*(CONVLNG0+RAD*COS(PI*SLAT(J)/180.)*PI/180.0)
        SLNG(J)=SLNG0-RA*COS(STHETA)/CONVLNG

        A1=((DLNG0+DCENTER*COS(ANGLE)/CONVLNG2)-SLNG(J))*CONVLNG2
        B1=(SLAT(J)-(DLAT0-DCENTER*SIN(ANGLE)/CONVLAT))*CONVLAT
        C1=SQRT(A1**2+B1**2)
        PSI=ATAN2(B1,A1)-THTA
        RIS=(C1*COS(PSI)/(DX))+1
        IS=INT(RIS)
        IF(IS.GT.NX) THEN
          WRITE(NPU,194)
          KSVP=0
          RETURN
        ENDIF

        RJS=(C1*SIN(PSI)/(DY))+1
        JS=INT(RJS)
        IF(JS.GT.NY) THEN
          WRITE(NPU,195)
          KSVP=0
          RETURN
```

```
      ENDIF

      IF(IS.LT.1) THEN
        WRITE(NPU,194)
        KSVP=0
        RETURN
      ENDIF

      IF(JS.LT.1) THEN
        WRITE(NPU,195)
        KSVP=0
        RETURN
      ENDIF

      XIP=C1*COS(PSI)
      YJP=C1*SIN(PSI)
      A2=DX*(IS-1)
      B2=DY*(JS-1)
      C2=SQRT(A2**2+B2**2)
      PSI2=ATAN2(B2,A2)
      XIS=C2*COS(PSI2)
      YJS=C2*SIN(PSI2)
      XOFF=(XIP-XIS)
      YOFF=(YJP-YJS)
      ZETA=ATAN2(YOFF,XOFF)
      GAMMA=PI-EPS-ZETA
      H1=SQRT(XOFF*XOFF+YOFF*YOFF)
      H=DXSEPS/SIN(GAMMA)
      I1=(JS-1)*NX+IS
      I2=I1+1
      I3=I1+NX
      I4=I1+NX+1
      IF(H1.LE.H) THEN
        DEPTH=BOT(I1)+XOFF*(BOT(I2)-BOT(I1))/DX+YOFF*
     +            (BOT(I3)-BOT(I1))/DY
      ELSE
        DEPTH=BOT(I4)+(DX-XOFF)*(BOT(I3)-BOT(I4))/DX+(DY-YOFF)*
     +            (BOT(I2)-BOT(I4))/DY
      ENDIF

      IF(J.EQ.1) DEPTH1=DEPTH
      XOFF=XOFF/DX
      YOFF=YOFF/DY

C     *** Interpolate
      DO LEV=1,NLEV
        P1=SVP(I1,LEV)
        P2=SVP(I2,LEV)
        P3=SVP(I3,LEV)
        P4=SVP(I4,LEV)
        A=(YOFF*P3+(1.0-YOFF)*P1)
        B=(YOFF*P4+(1.0-YOFF)*P2)
        CSVP(LEV,J)=(A*(1.0-XOFF)+B*XOFF)
        ZSVP(LEV,J)=ZLEV(LEV)
      ENDDO

      ILYR=1
```

```
         RHO(ILYR,J)=1.0
         RHOG(ILYR,J)=0.0
         BETA(ILYR,J)=0.0
         BETAG(ILYR,J)=0.0
         IF(DEPTH.GT.ZSVP(NLEV,J)) GO TO 140
         IF(DEPTH.EQ.ZSVP(NLEV,J)) GO TO 141

         DO L=1,NLEV
           LEV=L
           IF(DEPTH.EQ.ZSVP(LEV,J)) GO TO 138
           IF(DEPTH.LT.ZSVP(LEV,J)) GO TO 135
         ENDDO
         WRITE(NPU,*)'ERROR IN LOGIC. ???'
135      CONTINUE
         CSVP(LEV,J)=CSVP(LEV-1,J)+(CSVP(LEV,J)-CSVP(LEV-1,J))*
     +        (DEPTH-ZSVP(LEV-1,J))/(ZSVP(LEV,J)-ZSVP(LEV-1,J))
         ZSVP(LEV,J)=DEPTH
138      IXSVP(ILYR,J)=LEV
         ZLYR(ILYR,J)=DEPTH
         NSVP=LEV
         GO TO 145

140      ZSVP(NLEV+1,J)=DEPTH
         CSVP(NLEV+1,J)=CSVP(NLEV-1,J)+(CSVP(NLEV,J)-CSVP(NLEV-1,J))*
     +        (ZSVP(NLEV+1,J)-ZSVP(NLEV-1,J))/(ZSVP(NLEV,J)-ZSVP(NLEV-1,J))
         IXSVP(ILYR,J)=NLEV+1
         ZLYR(ILYR,J)=DEPTH
         NSVP=NLEV+1
         GO TO 145

141      IXSVP(ILYR,J)=NLEV
         ZLYR(ILYR,J)=ZSVP(NLEV,J)
         NSVP=NLEV

145      CONTINUE
         ILYR=2
         RHO(ILYR,J)=BOTRHO
         RHOG(ILYR,J)=BOTRHOG
C*       Next line commented out originally
C*     WRITE(NPU,*)'USVP BOTRHO,G ',BOTRHO,BOTRHOG
         ZG=ZLYR(ILYR,J)
         BETA(ILYR,J)=BOTBETA
         BETAG(ILYR,J)=BOTBETAG
         ZSVP(NSVP+1,J)=DEPTH
         CSVP(NSVP+1,J)=CSVP(NSVP,J)/CWCB
         IF(ZA-DEPTH.GE.SEDZ)THEN
           ZZ=SEDZ
         ELSE
           ZZ=ZA-DEPTH
         ENDIF

         CSVP(NSVP+2,J)=CSVP(NSVP+1,J)+(ZZ)*CGRAD
         ZSVP(NSVP+2,J)=DEPTH+ZZ
         ZLYR(ILYR,J)=DEPTH+ZZ
         IXSVP(ILYR,J)=NSVP+2
         NLYRS(J)=ILYR
       ENDDO
```

```
      NSVP=NSVP+2
      RAOLD=RA
      RETURN

190   WRITE(NPU,191)
191     FORMAT(1X,'DATA MISMATCH. NX*NY DOES NOT EQUAL NN.')
        NSVP=0
      RETURN

C     Get here from EOF read error
192   WRITE(NPU,193)
193     FORMAT(1X,'READ ERROR. CHECK INPUT SOUND SPEED FILE.')
        NSVP=0
      RETURN

194   FORMAT(1X,'LONGITUDE OUTSIDE LIMITS OF SOUND SPEED DOMAIN.')
195   FORMAT(1X,'LATITUDE OUTSIDE LIMITS OF SOUND SPEED DOMAIN.')

      END
```

APPENDIX E

KEY PROGRAMS USED IN THE FOR3D MODEL PROJECT

The purpose of this appendix is to provide documentation of the programs necessary to actually utilize the FOR3D Model. The author has spent considerable time developing these supporting codes. As they have been written in either FORTRAN 77 or Matlab version 5, they may have utility to others. All of these programs are compatible with the original output formats of the FOR3D Model with out any changes with the following exception. Only output stream formats generated with the input stream variable, DOUGRA=0, are supported. The complicating factor that results from DOUGRA $\neq$ 0 is that the output range step size is non-uniform. Code to deal with that condition has not been implemented.

The programs included in this appendix are listed in Table E1. Care should be taken in transcribing programs directly from this report as the word-processor automated word wrapping function may have mapped some code into incorrect columns.

Several other programs were developed for generating modified sound fields, generating and plotting bathymetry, and for generating the results for the 2D and 3D mode analogies presented in Sections 7 and 8 of the report. These programs were not considered of sufficient general utility for inclusion in the report but are available from the author.

Table E1

Program listings in Appendix E

E.1	**Preformatter Programs**
1.	harv_build.inc
2.	harv_build.f Reads files harvard.cfg and POM sound field. Outputs harvard.in and harvard.spd
E.2	**Plotting FOR3D Model Results**
1.	make_plot_files.f Reads File, HARVARD.OUT, with selection of radials to plot. Outputs plot file for Matlab input for each radial selected.
2.	plot_range_at_depth.m Reads in a radial plot file. Outputs a depth vs. range with a dB TL color scale “tiff” plot file.
3.	ovly_plots_range_at_depth.m Reads in two radial plot files, one each for the 2D and 3D modes. Overlays 2D and 3D Mode Outputs in a dB TL vs. range “tiff” plot file.
4.	ovly_plots_depth_at_range.m Reads in two radial plot files, one each for the 2D and 3D modes. Overlays 2D and 3D Mode Outputs in a dB TL vs. depth “tiff” plot file.
E.3	**Beamforming of FOR3D Model Results**
1.	make_beamer_file.f Reads in file HARVARD.OUT. Extracts data for beamforming to file beam.in for Matlab input.
2.	beamer.m Reads in file beam.in Outputs beamformed results to three “tiff” plots files: a. magnitude vs. wedge angle b. phase vs. wedge angle c. beam strength vs. beam angle.

Appendix E.1 Preformatter Programs

Program file 'harv_build.inc'

```
C*************** Beginning of harv_build.inc ***************************
      INTEGER NODIN,NODLOG
      PARAMETER(NODIN=15,NODLOG=16)
C        NODIN  = 15 is the ocean model file input I/O channel
C        NODLOG = 16 is the runtime log file channel

      INTEGER NFOR3D,NHV
      PARAMETER (NFOR3D=17,NHV=18)
C        NFOR3D = 17 is the harvard.in INPUT FILE
C        NHV    = 18 is the harvard formatted output file

      INTEGER NCFG
      PARAMETER(NCFG=19)
C        NCFG   = 19 is the configuration input file

      INTEGER*4 ODIXMX,ODIYMX,ODIZMX,MXY
      PARAMETER(ODIXMX=59 ,ODIYMX=79 ,ODIZMX=40,MXY=ODIXMX*ODIYMX)
C        ODIXMX  Max No elements in X direction
C        ODIYMX  Max No elements in Y direction, but last Y array
C                elements are dummy values, E+21 (shoreline)
C        ODIZMX  Max No elements in Z direction
C        The above should agree with first line of the ocean data file
C-----------------------------------------------------------------------
      COMMON /OCEAN/  ODIX,ODIY,ODIZ,
     +                FIRST_PASS_OCEAN,
     +                TDAY,DUM1,SOUND_IN,
     +                ODDX,ODDY,ODDZ,ODXMX,ODYMX,ODZMX

      INTEGER*4       ODIX,ODIY,ODIZ

      LOGICAL         FIRST_PASS_OCEAN

      REAL*4          TDAY,DUM1,SOUND_IN(ODIXMX,ODIYMX,ODIZMX),
     +                  ODDX,ODDY,ODDZ,ODXMX,ODYMX,ODZMX
C-----------------------------------------------------------------------
      COMMON /BUILD/  SLAT0,SLNG0,DLAT0,DLNG0,BOT,DX,DY,DZ,
     +                ZLEV,NX,NY,NZ,
     +                OKNO,
     +                ANSWER

      REAL*4          SLAT0,SLNG0,DLAT0,DLNG0,BOT(MXY),DX,DY,DZ,
     +                ZLEV(ODIZMX)

      INTEGER *4      NX,NY,NZ

      LOGICAL         OKNO

      CHARACTER       ANSWER*1
C-----------------------------------------------------------------------
      COMMON /CONFIG/ FRQ,ZS,C0,RA,ZA,FLDW,RMAX,DR,WDR,WZ1,
     +                WZ2,WDZ,WDTH,PDR,PDZ,PDTH,DOUGRA,
     +                U1,U2,U3,U4,U5,U6,U7,U8,U9,U10,U11,U12,
```

```
     +                     ISFLD,ISVP,IBOT,NDIV,N,IHNK,NDIM,ITYPES,
     +                     ITYPEB,ITYPPW,ITYPSW,NSEC,ISF,
     +                     TITLE

      REAL                 FRQ,ZS,C0,RA,ZA,FLDW,RMAX,DR,WDR,WZ1,
     +                     WZ2,WDZ,WDTH,PDR,PDZ,PDTH,DOUGRA,
     +                     U1,U2,U3,U4,U5,U6,U7,U8,U9,U10,U11,U12

      INTEGER*4            ISFLD,ISVP,IBOT,NDIV,N,IHNK,NDIM,ITYPES,
     +                     ITYPEB,ITYPPW,ITYPSW,NSEC,ISF

      CHARACTER            TITLE*80

C****************** Ending of harv_build.inc ***************************
```

Program file 'harv_build.f'

```
C*********************************************************************
C   File harv_build.f     Last Revised A. E. Leybourne  11/19/99
C*********************************************************************
C   This ROUTINE reads input files
C         harvard.in      (contains some of the run-time parameters
C         sound_in.dat    (the data grid from the ocean model)
C
C   Determines from run-time parameters what is needed in the output.
C
C   A constant or sloped bathymetry field as selected is generated
C   compatible in size with the sound speed field.  There is
C   a simple sloped bottom routine include which may be changed as
C   needed to accomodate different bottom topographies.
C
C   Then outputs file
C        harvard.spd
C
C   The file above is the same format as the sample e_ln3_f.spd that
C   was originally on the NJIT ftp download site.  Thus it should be
C   possible to process the file. If the for3d.f numeric model does not
C   get reasonable answers, the file formats should be verified first.
C*********************************************************************
C************** Beginning of harv_build.inc **************************
      INTEGER NODIN,NODLOG
      PARAMETER(NODIN=15,NODLOG=16)
C        NODIN  = 15 is the ocean model file input I/O channel
C        N0DLOG = 16 is the runtime log file channel

      INTEGER NFOR3D,NHV
      PARAMETER (NFOR3D=17,NHV=18)
C        NFOR3D = 17 is the harvard.in INPUT FILE
C        NHV    = 18 is the harvard formatted output file
      INTEGER NCFG

      PARAMETER(NCFG=19)
C        NCFG   = 19 is the configuration input file

      INTEGER*4 ODIXMX,ODIYMX,ODIZMX,MXY
      PARAMETER(ODIXMX=59 ,ODIYMX=79 ,ODIZMX=40,MXY=ODIXMX*ODIYMX)
C        ODIXMX  Max No elements in X direction
C        ODIYMX  Max No elements in Y direction, but last Y array
C                elements are dummy values, E+21 (shoreline)
C        ODIZMX  Max No elements in Z direction
C        The above should agree with first line of the ocean data file
C---------------------------------------------------------------------
      COMMON /OCEAN/  ODIX,ODIY,ODIZ,
     +                FIRST_PASS_OCEAN,
     +                TDAY,DUM1,SOUND_IN,
     +                ODDX,ODDY,ODDZ,ODXMX,ODYMX,ODZMX

      INTEGER*4       ODIX,ODIY,ODIZ

      LOGICAL         FIRST_PASS_OCEAN
```

```
      REAL*4             TDAY,DUM1,SOUND_IN(ODIXMX,ODIYMX,ODIZMX),
     +                     ODDX,ODDY,ODDZ,ODXMX,ODYMX,ODZMX
C-----------------------------------------------------------------------
      COMMON /BUILD/     SLAT0,SLNG0,DLAT0,DLNG0,BOT,DX,DY,DZ,
     +                   ZLEV,NX,NY,NZ,
     +                   OKNO,
     +                   ANSWER

      REAL*4             SLAT0,SLNG0,DLAT0,DLNG0,BOT(MXY),DX,DY,DZ,
     +                   ZLEV(ODIZMX)

      INTEGER *4         NX,NY,NZ

      LOGICAL            OKNO

      CHARACTER          ANSWER*1
C-----------------------------------------------------------------------
      COMMON /CONFIG/    FRQ,ZS,C0,RA,ZA,FLDW,RMAX,DR,WDR,WZ1,
     +                   WZ2,WDZ,WDTH,PDR,PDZ,PDTH,DOUGRA,
     +                   U1,U2,U3,U4,U5,U6,U7,U8,U9,U10,U11,U12,
     +                   ISFLD,ISVP,IBOT,NDIV,N,IHNK,NDIM,ITYPES,
     +                   ITYPEB,ITYPPW,ITYPSW,NSEC,ISF,
     +                   TITLE

      REAL               FRQ,ZS,C0,RA,ZA,FLDW,RMAX,DR,WDR,WZ1,
     +                   WZ2,WDZ,WDTH,PDR,PDZ,PDTH,DOUGRA,
     +                   U1,U2,U3,U4,U5,U6,U7,U8,U9,U10,U11,U12

      INTEGER*4          ISFLD,ISVP,IBOT,NDIV,N,IHNK,NDIM,ITYPES,
     +                   ITYPEB,ITYPPW,ITYPSW,NSEC,ISF

      CHARACTER          TITLE*80

C***************** Ending of harv_build.inc ***************************
      REAL*4             THTA
      LOGICAL            EXISTFLG
      CHARACTER          SOUND_FILE_IN*80,PATH*80
C-----------------------------------------------------------------------
C  Set path to the sound velocity data files
      PATH='/lhome/leybourn/for3d/3d_ocean/ocean_data/bob_fields/'
C  If harvard.cfg file exists open it else make a default sample file
      INQUIRE(FILE='harvard.cfg',EXIST=EXISTFLG)
      IF(EXISTFLG) THEN
        CALL READ_CONFIG (TITLE,NDIM,FRQ,ZS,C0,ISF,RA,ZA,N,IHNK,
     +                    ITYPES,ITYPEB,ITYPPW,ITYPSW,FLDW,NSEC,
     +                    RMAX,DR,WDR,WZ1,WZ2,WDZ,WDTH,PDR,PDZ,
     +                    PDTH,ISFLD,ISVP,IBOT,DOUGRA,NDIV,
     +                    U1,U2,U3,U4,U5,U6,U7,U8,U9,U10,U11,U12)
      ELSE
        CALL WRITE_CONFIG(TITLE,NDIM,FRQ,ZS,C0,ISF,RA,ZA,N,IHNK,
     +                    ITYPES,ITYPEB,ITYPPW,ITYPSW,FLDW,NSEC,
     +                    RMAX,DR,WDR,WZ1,WZ2,WDZ,WDTH,PDR,PDZ,
     +                    PDTH,ISFLD,ISVP,IBOT,DOUGRA,NDIV,
     +                    U1,U2,U3,U4,U5,U6,U7,U8,U9,U10,U11,U12)
        PRINT *
        PRINT *,'A default run configuration file has just been created'
        STOP 'You should now edit it for your run time configuration'
```

```
      ENDIF

C    Latitude of starting field
      SLAT0=U2
C    Longitude of starting field
      SLNG0=U3
C    Direction of center ray, degrees CW from North
      DIR=U4

C   Set default data domain CENTER location
      DLNG0=40.69
      DLAT0=65.32

C   Get location CENTER of the sound field domain
      PRINT *,'You will be prompted for data identifing the location of'
      PRINT *,'Your sound speed file data domain if you do not accept'
      PRINT *,'the default locations. Note W is +, North is +.'
      PRINT *
      PRINT *,'Default domain CENTER Longitutude  = ',DLNG0
      PRINT *,'                      Latitude     = ',DLAT0
      PRINT *
      PRINT *, 'Accept DefaultS    (Y/N) or Return'
      CALL YESNO(OKNO)
      IF(OKNO) THEN
        PRINT *
        PRINT *, ' Enter LONGITUDE, LATITUDE'
        ACCEPT *, DLNG0,DLAT0
      ENDIF

      PRINT *
      PRINT *,'Enter sound input filename in subdirectory',
     +       '       ./bob_fields to process, 75 char. max.'
      ACCEPT 90, SOUND_FILE_IN
90    FORMAT(A)
      CALL CLEN(PATH,80,N1CHAR)
C     print *,'*'//path(1:n1char)//'*',n1char
      CALL CLEN(SOUND_FILE_IN,80,N2CHAR)
      INQUIRE(FILE=PATH(1:N1CHAR)//SOUND_FILE_IN(1:N2CHAR),
     +        EXIST=EXISTFLG)

      IF( .NOT. EXISTFLG) THEN
        PRINT *
        PRINT *,'File requested :'
        PRINT *
        PRINT *,PATH(1:N1CHAR)//SOUND_FILE_IN(1:N2CHAR)
        PRINT *,'   was not found -- you must firstcopy it to',
     +          'directory:'
        PRINT *
        PRINT *,PATH
        PRINT *
        STOP
      ENDIF

C    Open files
      OPEN(NODIN, FILE=PATH(1:N1CHAR)//SOUND_FILE_IN(1:N2CHAR),
     +       STATUS='OLD',FORM='FORMATTED')
```

```
      OPEN(NODLOG, FILE='ocean_dat.log',STATUS='UNKNOWN',
     +       FORM='FORMATTED')

      OPEN(UNIT=NHV,FILE='harvard.spd',STATUS='UNKNOWN',
     +                        FORM='FORMATTED')

C  Get all information except sound speed from the ocean model data file
      FIRST_PASS_OCEAN = .TRUE.
      CALL READ_OCEAN_FILE

C   Subroutine call provides user with SV data grid extent in
C   latitude/longitude prior to query about desired source location
C   and does some otherchecks on domain validity.
      CALL LAT_LONG(SLAT0,SLNG0,DX,NX,DY,NY,DLAT0,DLNG0,THTA,
     +              RMAX,DIR,FLDW,NSEC)

C   Update Latitude of starting field
      U2=SLAT0
C   Update Longitude of starting field
      U3=SLNG0


C   Note: Some of the data in the configuration file is revised because
C   the interactive user inquiries prior to writing out this file. Just
C   prior to this write is a good place to place any additional user
C   interactive queries.

      CALL WRITE_HARVARD_IN(TITLE,NDIM,FRQ,ZS,C0,ISF,RA,ZA,N,IHNK,
     +                       ITYPES,ITYPEB,ITYPPW,ITYPSW,FLDW,NSEC,
     +                       RMAX,DR,WDR,WZ1,WZ2,WDZ,WDTH,PDR,PDZ,
     +                       PDTH,ISFLD,ISVP,IBOT,DOUGRA,NDIV,
     +                       U1,U2,U3,U4,U5,U6,U7,U8,U9,U10,U11,U12)
      PRINT *

C  Build FOR3D compatible file 'harvard.spd', Info for Lines 1 thru 5
C  were read during first CALL to READ_OCEAN_FILE

C     WRITE Statement 1, into harvard format:

        WRITE(NHV,103)DLAT0,DLNG0,THTA,NX,NY
103     FORMAT(2X,2F7.3,F8.5,2I6)
C             DLATO  Latitude of domain center
C             DLNGO  Longitude of domain center
C             THTA   CCW domain rotation in radians, .i.e., CCW Angle
C                    array X axis to a line parallel to the equator
C             NX     No steps in X direction
C             NY     No steps in Y direction

C     WRITE Statement 2, into harvard format:
C------------------------------------------------------------------------
        WRITE(NHV,104)DX,DY,NZ
104     FORMAT(2(F7.1),I6)
C             DX     Increment X direction, kmeters
C             DY     Increment Y direction, kmeters
C                    (After input these are converted to meters)
C             NZ   No of levels in the file
C                    and   NZ .LE. MXLEX
```

```
C     WRITE StatementS 3, into harvard format:
C-----------------------------------------------------------------------
        WRITE(NHV,105) (ZLEV(ILEV),ILEV=1,NZ)
105     FORMAT(10F7.1)
C             ZLEV   Values for each data level
C             Note the sucess of this write depends on each record
C             containing 10 data fields, except the last record which
C             can be only partially filled; since Fortran reuses the
C             FORMAT repetitively until the ouput list is satisfied.

C     WRITE Statement 4, into harvard format:
C-----------------------------------------------------------------------
        NN=NX*NY
        T=0
        ITYP=0
        KK=0
        ITMP1=0
        ITMP2=0
        WRITE(NHV,7)NN,T,ITYP,KK,ITMP1,ITMP2
7       FORMAT(2X,I5,F10.5,4I5)
C             NN     No. elements at each level
C                    (NN must = NXY
C             T      Not used
C             ITYP   Not used
C             KK     Not used
C             ITMP1  Not used
C             ITMP2  Not used

C     WRITE Statement 5, into harvard format:
C-----------------------------------------------------------------------
        TEMP1=(NZ-1)*ODDZ
C     Query for type of Bottom
        PRINT *,'      What type bathymetry do you wish to have ?'
        PRINT *
        PRINT *,'1      Constant bathymetry', TEMP1,' meters'
        PRINT *
        PRINT *,'2      Diagonal slope:'
        PRINT *,'          ',0.5*TEMP1,' at (0,0)'
        PRINT *,'          ',0.75*TEMP1,' at (0,NY) and (NX,0)'
        PRINT *,'          ',TEMP1,' at (NX,NY)'
        PRINT *
        PRINT *,'3      Slope',0.5*TEMP1,'at X=0', TEMP1,' at X=Xmax'
        PRINT *
        PRINT *, 'Enter you selection number'
        ACCEPT *,ISELECT
        GOTO(10,20,30) ISELECT
        STOP 'Menue selecton error'

C     The number of RECORDS generated may vary
C     Fill the whole array with the same bottom depth
10      DO I=1,NN
          BOT(I)=TEMP1
        ENDDO
        GOTO 100

C     Fill the array with a sloped bottom
```

```
20        PRINT *,'Generating a Diagonally sloped bottom'
          DO J=1,NY
            DO I=1,NX
              K=(J-1)*NX+I
              BOT(K)=0.25*TEMP1*(2.0+FLOAT(J-1)/(NY-1)+FLOAT(I-1)/(NX-1))
            ENDDO
          ENDDO
          GOTO 100

C       Fill the array with a sloped bottom
30        PRINT *,'Generating a LEFT to RIGHT sloped bottom'
          DO J=1,NY
            DO I=1,NX
              K=(J-1)*NX+J
              BOT(K)=0.5*TEMP1*(1.0+FLOAT(I-1)/(NX-1))
            ENDDO
          ENDDO
          GOTO 100

C         Write out to the file
100         WRITE(NHV,108)(BOT(I),I=1,NN)
108         FORMAT(8F10.3)

C         Now get the sound speed data, file is already positioned there
            CALL READ_OCEAN_FILE

C       WRITE Statement 6, into harvard format:
C-----------------------------------------------------------------------
C         WRITE NZ levels of speed data
          DO K=1,NZ
            WRITE(NHV,7)NN,T,ITYP,KK,ITMP1,ITMP2
            WRITE(NHV,FMT=110) ((SOUND_IN(I,J,K),I=1,NX),J=1,NY)
          ENDDO
110     FORMAT(5(E12.6))

C     Close all files
        CLOSE(NODIN)
        CLOSE(NODLOG)
        CLOSE(NFOR3D)
        CLOSE(NHV)
        PRINT *
        PRINT *,'New file "harvard.spd" created OK'
        PRINT *,'New file "harvard.in"  created OK'
        PRINT *
        STOP 'Normal program exit'

        END

        SUBROUTINE READ_OCEAN_FILE
C**********************************************************************
C     If FIRST_PASS_OCEAN is .TRUE., all info from the file is read in
C     except sound speed data.  If FIRST_PASS_OCEAN is .FALSE. sound speed
C     is read in and written back out to reformatted file
C**********************************************************************
C     As provided there is no data in the file to enable the user to tell
C     how this model data grid is located and oriented on the earth.
C     The file simply has an X,Y,Z =(0,0,0) beginning location.
```

```
C
C   All distances are in meters, with Z measured from the surface.
C        ODXMX=(ODIX-1)*ODDX,   ODYMX=(ODIY-1)*ODDY,   ODZMX=(ODIZ-1)*ODDZ
C
C      ABOUT THE DATA (as per Germana Peggion)
C
C             The data were generated with a coriolis parameter
C                 that is between 35 and 55 degree. The coastline is
C                 parallel to the equator.
C
C             Bob Fields has a copy of the theoretical/numerical paper
C                 on which the data is based.
C
C             The upper side (j=odiy) is supposed to be the coast
C             Values stored there are fake data 0.10E+21, not sound speed
C
C             Flat bottom assumption was used
C                 H=ODDZ*FLOAT(ODIZ-1)
C
C            The domain matrix is:
C
C                        ODIX,ODIY
C            --------------
C           |              |
C           |              |       where ODDZ is the depth increment
C           |              |       down from the surface
C            --------------
C        1,1
C
C*********************************************************************
      INCLUDE 'harv_build.inc'
C---------------------------------------------------------------------
      IF (FIRST_PASS_OCEAN) THEN
C     Open ocean acoustic model data file and log file

C     Read data line 1 and check for file size compatability
        READ(NODIN,91, ERR=999) ODIX,ODIY,ODIZ
91      FORMAT(3I6)

        IF(ODIX.GT.ODIXMX .OR. ODIY.GT.ODIYMX .OR.ODIZ.GT.ODIZMX) THEN
          PRINT 92, ODIXMX,ODIX,ODIYMX,ODIY,ODIZMX,ODIZ
          WRITE(NODLOG,92) ODIXMX,ODIX,ODIYMX,ODIY,ODIZMX,ODIZ
92        FORMAT('PARAMETERS:'/
     +        T10,'ODIXMX=',I6,'  Must be >=',I6/
     +        T10,'ODIYMX=',I6,'  Must be >=',I6/
     +        T10,'ODIZMX=',I6,'  Must be >=',I6//)
          STOP 'Fix and recompile'
        ENDIF

        PRINT *, 'Some of the values read from the ocean_model file'
        PRINT *
        PRINT 94,ODIX,ODIY,ODIZ
        WRITE(NODLOG,94) ODIX,ODIY,ODIZ
94      FORMAT ('      ODIX        ODIY         ODIZ'/,I8,2(3x,I8)/)

C     Read data line 2
C         TDAY=Time in days  (Unused value)
```

```
C          DUM1=Land points (undefined value)
C          ODDX      input grid increments
C          ODDY         in meters
C          ODDZ
         READ(NODIN,100,ERR=999) TDAY,DUM1,ODDX,ODDY,ODDZ
 100     FORMAT(5(E12.6))
         PRINT 101,TDAY,ODDX,ODDY,ODDZ
         WRITE(NODLOG,101) TDAY,ODDX,ODDY,ODDZ
 101     FORMAT ('     TDAY       ODDX        ODDY        ODDZ'/,
     +           F9.3,3(1X,F10.3)/)

C       Several variables below, for compatibility with for3d_all_ocean.f
         NX=  ODIX
C        The last Y array elements are not SV data, but dummy values,
C         0.1 E+21,indication of shoreline at that grid point
         NY=  ODIY-1
         NZ=ODIZ

C       In meters
         ODXMX=(NX-1)*ODDX
         ODYMX=(NY-1)*ODDY
         ODZMX=(NZ-1)*ODDZ

C        In kilometers
           DX=ODDX/1000.
           DY=ODDY/1000.

C       Make the levels evenly spaced
         DO ILEV=1,NZ
           ZLEV(ILEV)=(ILEV-1)*ODDZ
         ENDDO

         PRINT 102,ODXMX,ODYMX,ODZMX
         WRITE(NODLOG,102) ODXMX,ODYMX,ODZMX
 102     FORMAT('    ODXMX      ODYMX       ODZMX or depth'/,3F11.1/)
         FIRST_PASS_OCEAN =.FALSE.
         RETURN
       ENDIF

       IF (.NOT. FIRST_PASS_OCEAN) THEN
C      Read sound speed data into the input array SOUND_IN
         READ(UNIT=NODIN,FMT=105, ERR=999)
     +       (((SOUND_IN(I,J,K),I=1,ODIX),J=1,ODIY),K=1,ODIZ)
 105     FORMAT(12(E12.6))
         PRINT *
         PRINT *, 'Ocean Model sound speed grid read OK'
         RETURN
       ENDIF

 999   WRITE (NODLOG,FMT=*) 'There was a problem reading the input',
     +                      'data file'
       STOP 'There was a problem reading the input data file'
       END


       SUBROUTINE LAT_LONG(SLAT0,SLNG0,DX,NX,DY,NY,DLAT0,DLNG0,THTA,
     +                RMAX,DIR,FLDW,NSEC)
```

```
C**********************************************************************
C   Displays extent of the ocean model sound speed in Lat/Long for
C   assistance to the user when picking the source location.
C
C   Central to the success of this routine, the ocean model data must
C   be presented so that
C         Longitude - value is East,    + value is West
C         Latitude  + value is North,   - value is South
C         DIR is propagation degrees CW from North of central solution
C         THTA is X axis radians CCW from line parallel to the equator
C
C   The validity of the extreme range of tracks is also checked prior
C   to creating the required FOR3D model files.
C**********************************************************************
      REAL*4    CONVLNGDC,DLNG0,LNGLL,LNGLR,LNGUL,LNGUR
      REAL      CONVLATDC,DLAT0,LATLL,LATLR,LATUL,LATUR
      REAL*4    SLAT0,SLNG0,DX,DY,THTA,RMAX,DIR,FLDW
      INTEGER*4 NX,NY,NSEC
      LOGICAL OKNO,DOMAIN_OK
C----------------------------------------------------------------------
C   Most variables above are as defined in the config file routine
C----------------------------------------------------------------------
      PI=4*ATAN(1.0)
C   Approximate radius of earth (as used in FOR3D model)
      RAD=6378400

      ABS_DLAT0=ABS(DLAT0)
      IF(ABS_DLAT0 .GT. 85) THEN
        PRINT *, 'Variable DLAT0 is out of range'
      ENDIF

      ABS_DLNG0=ABS(DLNG0)
      IF(ABS_DLNG0 .GT. 180) THEN
        PRINT *, 'Variable DLNG0 is out of range'
      ENDIF

C   In meters
      XBASIN=1000.*DX*(NX-1)
      YBASIN=1000.*DY*(NY-1)

C   In kilometers
      SIDEX=XBASIN/1000.
      SIDEY=YBASIN/1000.

C   Get degrees longitude/meter at Domain Center
      CONVLNGDC=ABS(RAD*COS(PI*DLAT0/180.)*PI/180.)

C   Get meter/degrees latitude at Domain Center
      CONVLATDC=PI*RAD/180.

C   Check status of default source location
      ISOL=0
      CALL CHECK_DOMAIN(ISOL,XBASIN,YBASIN,DLNG0,DLAT0,DX,DY,THTA,
     +             SLNG0,SLAT0,CONVLNGDC,CONVLATDC,NX,NY,DOMAIN_OK)

C   Distance, grid origin to center
      DCENTER=SQRT((XBASIN/2.)**2 + (YBASIN/2.)**2)
```

```
C   Angle for X axis to center diagonal, in radians
      CENTANG=ATAN2(YBASIN,XBASIN)
C   Angle from line parallel to equator to grid diagonal, in radians
      ANGLE=CENTANG+THTA
C   Corners of the Domain
      AA=     DCENTER*COS(ANGLE)
      BB=     DCENTER*SIN(ANGLE)
      LNGUR= DLNG0-AA/CONVLNGDC
      LNGLL= DLNG0+AA/CONVLNGDC
      LATUR= DLAT0+BB/CONVLATDC
      LATLL= DLAT0-BB/CONVLATDC

      AA=     YBASIN*COS(THTA+PI/2.)
      BB=     YBASIN*SIN(THTA+PI/2.)
      LNGUL= LNGLL+AA/CONVLNGDC
      LATUL= LATLL+BB/CONVLATDC
      AA=     XBASIN*COS(THTA)
      BB=     XBASIN*SIN(THTA)
      LNGLR= LNGLL-AA/CONVLNGDC
      LATLR= LATLL+BB/CONVLATDC

      PRINT *
      PRINT 30
30    FORMAT(50('-'))
      PRINT *,'To visualize grid, rotate it CCW ',THTA*180/PI,' degrees'
      PRINT 30
      PRINT *
      PRINT 40,LNGUL,LNGUR,' Lng'
40    FORMAT ('      ',F8.4,'                    ',F8.4,A)
      PRINT 40,LATUL,LATUR,' Lat'
      PRINT *,'            -----------------------'
      PRINT *,'          ^ |                      |'
      PRINT *,'          | |                      |'
      PRINT *,'          Y |                      |'
      PRINT *,'            |                      |'
      PRINT 42,DLNG0
42    FORMAT('             |     ',F8.4,' Lng     |')
      PRINT 46,DLAT0,SIDEY
46    FORMAT('             |     ',F8.4,' Lat     | ',F8.3,' km')

      PRINT *,'            |                      |'
      PRINT *,'            |                      |'
      PRINT *,'            |                      |'
      PRINT 48, SIDEX
48    FORMAT('             |  ',F8.3' km          |')
      PRINT *,'            -----------------------   X -->'
      PRINT 40, LNGLL,LNGLR,' Lng'
      PRINT 40, LATLL,LATLR,' Lat'
      PRINT *
      PRINT *
      PRINT 30
      PRINT *
      PRINT *

      IF(DOMAIN_OK) THEN
C        Query user for location of the source latitude, longitude
C        Defaults defined in the configuration file, U2,U3
```

```
      PRINT *,'BEFORE RESPONDING TO QUERY, PLEASE STUDY GRID ABOVE'
      PRINT *
      PRINT *, 'Default  source location Lng/Lat',SLNG0,SLAT0
      PRINT *
      PRINT *, 'Accept Defaults    (Y/N) or Return'
      CALL YESNO(OKNO)
      IF(OKNO) THEN
        PRINT *
        PRINT *, 'Enter source longitude, latitude'
        PRINT *, '        East is -,  North is +'
        PRINT *, '        West is +,  South is -'
        PRINT *
        ACCEPT * ,SLNG0,SLAT0

C       Check status of source locations just entered
        CALL CHECK_DOMAIN(ISOL,XBASIN,YBASIN,DLNG0,DLAT0,DX,DY,THTA,
     +             SLNG0,SLAT0,CONVLNGDC,CONVLATDC,NX,NY,DOMAIN_OK)
      ENDIF
      PRINT *
     ENDIF

    DO WHILE(.NOT. DOMAIN_OK)
C     Query user for location of the source latitude, longitude
      PRINT *,'Source location entered not in the domain'
      PRINT *
      PRINT *, 'Enter source longitude, latitude'
      PRINT *, '        East is -,  North is +'
      PRINT *, '        West is +,  South is -'
      PRINT *
      ACCEPT * ,SLNG0,SLAT0
      CALL CHECK_DOMAIN(ISOL,XBASIN,YBASIN,DLNG0,DLAT0,DX,DY,THTA,
     +           SLNG0,SLAT0,CONVLNGDC,CONVLATDC,NX,NY,DOMAIN_OK)
      PRINT *
    ENDDO

C   Check to see if end of tracks are in the domain
    PRINT *,'Checking to see if end of tracks are in the domain'

    NSOLS=NSEC+1
C   Width of sector in degrees
    DTH=FLDW/NSEC
    DO ISOL=1,NSOLS
      PRINT *
      STHETA=(90.-DIR+FLDW/2.-(ISOL-1)*DTH)*PI/180.
      PRINT 60,'ISOL=',ISOL,'  STHETA= ',STHETA,' Rad.  STHETA= ',
     *        STHETA*(180/PI),' Deg.'
60    FORMAT(A,I4,A,F10.8,A,F10.6,A)
      TSTLNG=SLNG0-RMAX*COS(STHETA)/CONVLNGDC
      TSTLAT=SLAT0+RMAX*SIN(STHETA)/CONVLATDC

      CALL CHECK_DOMAIN(ISOL,XBASIN,YBASIN,DLNG0,DLAT0,DX,DY,THTA,
     +           TSTLNG,TSTLAT,CONVLNGDC,CONVLATDC,NX,NY,DOMAIN_OK)
      IF(.NOT. DOMAIN_OK) THEN
        PRINT *,'End of track for solution ',ISOL,' not in domain'
        STOP
      ENDIF
    ENDDO
```

```
      PRINT *
      PRINT *,'End of track CHECKS are OK'

      RETURN
      END

      SUBROUTINE CHECK_DOMAIN(ISOL,XBASIN,YBASIN,DLNG0,DLAT0,DX,DY,THTA,
     +               TSTLNG,TSTLAT,CONVLNGDC,CONVLATDC,NX,NY,DOMAIN_OK)
C***********************************************************************
C   This subroutine tests to see if location is in the data domain
C***********************************************************************
      REAL*4 DLNG0,DLAT0,DX,DY,THTA,TSTLNG,TSTLAT,CONVLNGDC,CONVLATDC
      REAL*4 DELMLNG,DELMLAT,DELHYP,DELX,DELY,PSI
      INTEGER*4 NX,NY
      LOGICAL DOMAIN_OK
C-----------------------------------------------------------------------
      PI=4*ATAN(1.0)
      DOMAIN_OK=.TRUE.
      ABS_TSTLAT=ABS(TSTLAT)
      IF(ABS_TSTLAT .GT. 90) THEN
        PRINT *, 'Variable TSTLAT is out of range'
        DOMAIN_OK=.FALSE.
      ENDIF

      PRINT 40 ,'TSTLNG = ',TSTLNG,'   TSTLAT=  ',TSTLAT
40    FORMAT(10X,2(A,F10.5))

      DELMLNG=(DLNG0-TSTLNG)*CONVLNGDC
      DELMLAT=(TSTLAT-DLAT0)*CONVLATDC
      DELHYP=SQRT(DELMLNG**2.+DELMLAT**2)
      IF(DELHYP .LT. 1.0E-10) THEN
        DELX=0
        DELY=0
      ELSE
        PSI=ATAN2(DELMLAT,DELMLNG)-THTA
        DELX=DELHYP*COS(PSI)
        DELY=DELHYP*SIN(PSI)
      ENDIF
      XMLOC=XBASIN/2. + DELX
      YMLOC=YBASIN/2. + DELY
      PRINT 80,'XLOC = ',XMLOC,'   YLOC = ',YMLOC

80    FORMAT(10X,2(A,F12.2))
      IF(XMLOC .LT. 0) THEN
        PRINT *,'XLOC Too Low'
        DOMAIN_OK=.FALSE.
      ENDIF

      IF(XMLOC .GT. XBASIN) THEN
        PRINT *,'XLOC Too High'
        DOMAIN_OK=.FALSE.
      ENDIF

      IF(YMLOC .LT. 0) THEN
        PRINT *,'YLOC Too Low'
        DOMAIN_OK=.FALSE.
      ENDIF
```

```
      IF(YMLOC .GT. YBASIN) THEN
        PRINT *,'YLOC Too High'
        DOMAIN_OK=.FALSE.
      ENDIF

      RETURN
      END

      SUBROUTINE WRITE_CONFIG(TITLE,NDIM,FRQ,ZS,C0,ISF,RA,ZA,N,IHNK,
     +                        ITYPES,ITYPEB,ITYPPW,ITYPSW,FLDW,NSEC,
     +                        RMAX,DR,WDR,WZ1,WZ2,WDZ,WDTH,PDR,PDZ,
     +                        PDTH,ISFLD,ISVP,IBOT,DOUGRA,NDIV,
     +                        U1,U2,U3,U4,U5,U6,U7,U8,U9,U10,U11,U12)
C*********************************************************************
C   This subroutine generates a starting 'harvard.cfg' file if one does
C   not exist, annotated so that one can identify the variables.
C   The user should make changes to 'harvard.cfg' to control the run.
C*********************************************************************
      INTEGER NCFG
      PARAMETER (NCFG=19)
      CHARACTER TITLE*80
C*********************************************************************
      OPEN(NCFG,FILE='harvard.cfg',STATUS='UNKNOWN',FORM='FORMATTED')
      TITLE=
     + 'Default Configuration file, discussion with R. Fields 12/7/98'
      WRITE(NCFG,90) TITLE
90      FORMAT(T10,A/)
      WRITE(NCFG,92)
92      FORMAT(T10,'If the file: harvard.cfg does not exist, one is'/
     +         T25,'generated for you.'//
     +         T10,'You may change the defaults by editing the subr.,'/
     +         T10,' WRITE_CONFIG.  Do not change the line spacings',
     +             '!!'//
     +         T10,'It is intended that you edit the configuration'/
     +         T10,'file to set up the parameters for your run.'//
     +         T10,'At the present time, the only configuration that'/
     +         T10,'  has been tested is the harvard test set that'/
     +         T10,'   was downloaded from the NJIT models site'//
     +         T10,'NOTE: ALL COMBINATIONS OF SWITCHES MAY NOT RUN'/)

      NDIM=3
      WRITE(NCFG,100) NDIM
100     FORMAT(I10,T20,'NDIM'/
     +         T20,'  1 - Model generates 2D solution'/
     +         T20,'  2 - Model generates NSOL x 2D solutions'/
     +         T20,'  3 - Model generates NSOL, 3D solutions'/)

      FRQ=1000
      WRITE(NCFG,102) FRQ
102     FORMAT(F10.2,T20,'FRQ, Frequency, Hz'/)

      ZS=100
      WRITE(NCFG,104) ZS
104     FORMAT(F10.2,T20,'ZS, Source depth - meters'/)

      C0=1500.0
```

```
      WRITE(NCFG,106) C0
106     FORMAT(F10.2,T20,'C0, Reference speed of sound - meters/sec'/
    +          T20'  If C0=0, C0 is set to avg. of first layer'/)

      ISF=2
      WRITE(NCFG,108) ISF
108     FORMAT(I10,T20,'ISF Starting field flag'/
    +         T20,'  0 - Program generates gaussian starting field'/
    +         T20,'      at range = 0.0. see SUBROUTINE SFLD3D.'/
    +         T20,'  1 - User supplies starting field. see SUBR.',
    +                 ' USFLD3D'/
    +         T20,'  2 - Greens wide angle starter'/
    +         T20,'  3 - SPARE'/)

      RA=0
      WRITE(NCFG,110) RA
110     FORMAT(F10.2,T20,'RA Horizontal range, from source to starting'/
    +          T20,'  field - meters.'/
    +          T20,'     RA is set to 0.0 if starting field is',
    +              ' Gaussian.'/
    +          T20,'     RA is incremented by DR as solution is ',
    +              'marched /
    +          T20,'       out in range.'/)

      ZA=195
      WRITE(NCFG,112) ZA
112     FORMAT(F10.2,T20,'ZA  Depth of field at range RA - meters'/
    +         T20,'  If ZA=0, set ZA to max depth of bot. layer 1st
profile'//
    +         T20,'  Else, initial depth of starting field at range RA
is:'/
    +         T20,'    If ITYPEB = 0 or 1, set ZA to maximum depth of'/
    +         T20,'      bottom-most sediment layer at initial range
of'/
    +         T20,'      starting field.'/
    +         T20,'    If ITYPEB = 3, ZA is maximum depth of
artificial'/
    +         T20,'      absorbing layer at initial range of starting
field.'/
    +         T20,'      program inserts layer.'/
    +         T20,'        RHO and BETA are obtained from layer above.'/
    +         T20,'        speed is bottom-most speed from layer
above.'/
    +         T20,'        bottom of absorbing layer remains flat.'/)

      N=1999
      WRITE(NCFG,114) N
114     FORMAT(I10,T20,'N Number of equi-spaced receivers in U'/
    +         T20,'  U is array - complex acoustic pressure field.'/
    +         T20,'  includes bottom point - but not surface point.'/
    +         T20,'  If N=0, N is computed at 1/10 of wavelength.'/)

      IHNK=0
      WRITE(NCFG,116) IHNK
116     FORMAT(I10,T20,'IHNK  - Hankel function flag'/
    +         T20,'  0 - Hankel fun. not used. 10*log(R) added'/
    +         T20,'      to solution.'/
```

```
     +         T20,'  1 - Starting field divided by Hankel function.'/
     +         T20,'      solution multiplied by Hankel function before'/
     +         T20,'      computing propagation loss.'/)

      ITYPES=0
      WRITE(NCFG,117) ITYPES
117     FORMAT(I10,T20,'ITYPES - Type of surface'/
     +         T20,'  0 - Press. release. SCON3D sets SURY and SURX = ',
     +                '0.0'/
     +         T20,'  1 - User supplies surface condition.  SUBR.',
     +                'USCON3D'/
     +         T20,'  2 - SPARE'/)

      ITYPEB=3
      WRITE(NCFG,118) ITYPEB
118     FORMAT(I10,T20,'ITYPEB - Type of bottom'/
     +         T20,'  0 - Pressure release, BCON3D sets BOTY and BOTX =
0.0'/
     +         T20,'  1 - User bottom condition. see SUBR. UBCON3D.'/
     +         T20,'  2 - SPARE.'/
     +         T20,'  3 - Absorbing layer used - bottom of layer is',
     +                 ' flat'/
     +         T20,'  4 - SPARE.'/)

      ITYPPW=2
      WRITE(NCFG,120) ITYPPW
120     FORMAT(I10,T20,'ITYPPW - Type port sidewall boundary cond.'/
     +         T20,'  0 - Field along port sidewall is set to 0.0.'/
     +         T20,'  1 - User supplied. See SUBROUTINE UPORT3D.'/
     +         T20,'  2 - Model generates 2D Solution if NDIM = 3.'/)

      ITYPSW=2
      WRITE(NCFG,122) ITYPSW
122     FORMAT(I10,T20,'ITYPSW - Type of stbd sidewall boundary cond.'/
     +         T20,'  0 - Field along stbd sidewall is set to 0.0.'/
     +         T20,'  1 - User supplied. See SUBROUTINE USTBD3D.'/
     +         T20,'  2 - Model generates 2D Solution if NDIM = 3.'/)

      FLDW=20.0
      WRITE(NCFG,124) FLDW
124     FORMAT(F10.2,T20,'FLDW  Width of field, degrees. Ignored if
NDIM=1'/)

      NSEC=4
      WRITE(NCFG,126) NSEC
126     FORMAT(I10,T20,'NSEC Number of sectors in field. Ignored if
NDIM=1'/
     +          T20,'  Number of solutions, NSOL=NSEC+1'/)

      RMAX=60000
      WRITE(NCFG,130)RMAX
130     FORMAT(F10.2,T20,'RMAX Maximum range of solution - meters.'/)

      DR=0.1
      WRITE(NCFG,132) DR
132     FORMAT(F10.2,T20,'DR  Range step - meters'/
```

```
     +          T20,'  If DR = 0, DR is set to 1 meter, then if bottom
is'/
     +          T20,'  not flat, DR is recomputed so that max depth is '/
     +          T20,'  incremented or decremented by DZ'/)

      WDR=30
      WRITE(NCFG,134) WDR
134     FORMAT(F10.2,T20,'WDR Range step solution is output - meters'/
     +          T20,'  WDR is rounded to nearest DR'/)

      WZ1=1
      WRITE(NCFG,136) WZ1
136     FORMAT(F10.2,T20,'WZ1 First receiver depth solution is output'/)

      WZ2=183
      WRITE(NCFG,138) WZ2
138     FORMAT(F10.2,T20,'WZ2 Last receiver depth solution is output'/
     +          T20,'  In other words, write WZ1 to WZ2 by WDZ - Meters'/)

      WDZ=.98
      WRITE(NCFG,140)WDZ
140     FORMAT(F10.2,T20,'WDZ Depth step solution is output - meters'/
     +          T20,'  WDZ  Selected so that plot program does not '/
     +          T20,'  interpolate between widely spaced receivers.'/
     +          T20,'  WDZ rounded to nearest DZ'/)

      WDTH=5
      WRITE(NCFG,142) WDTH
142     FORMAT(F10.2,T20,'WDTH Azimuthal step soln. is output - deg'/
     +          T20,'  WDTH rounded to nearest DTH'/)

      PDR=1000
      WRITE(NCFG,144) PDR
144     FORMAT(F10.2,T20,'PDR Range incr. for soln. output - meters'/
     +          T20,'  PDR rounded to nearest DR'/)

      PDZ=20.97
      WRITE(NCFG,146) PDZ
146     FORMAT(F10.2,T20,'PDZ Depth incr. for soln. output  -',
     +          ' meters'/
     +          T20,'  PDZ rounded to nearest DZ'/)

      PDTH=5
      WRITE(NCFG,148) PDTH
148     FORMAT(F10.2,T20,'PDTH  - Azimuthal incr. for soln. output -',
     +          ' degrees.'/
     +          T20,'  PDTH rounded to nearest DTH'/)

      ISFLD=1
      WRITE(NCFG,150) ISFLD
150     FORMAT(I10,T20,'ISFLD - Starting field print flag'/
     +          T20,'  0 - Do not print starting field'/
     +          T20,'  1 - Print starting field'/)

      ISVP=0
      WRITE(NCFG,152) ISVP
152     FORMAT(I10,T20,'ISVP  - SVP print flag'/
```

```
     +        T20,'  0 - Do not print sound velocity profile'/
     +        T20,'  1 - Print sound velocity profile'/)

      IBOT=0
      WRITE(NCFG,154). IBOT
154     FORMAT(I10,T20,'IBOT Bottom depth print flag'/
     +        T20,'  0 - Do not print bottom depths'/
     +        T20,'  1 - Print bottom depths'/)

      DOUGRA=5000
      WRITE(NCFG,156) DOUGRA
156     FORMAT(F10.2,T20,'DOUGRA  range flag'/
     +        T20,'  0  Use Crank-Nicolson method'/
     +        T20,' a #, Is the range to switch to the Douglas',
     +             ' method - meters'/)

      NDIV=5
      WRITE(NCFG,158) NDIV
158     FORMAT(I10,T20,'NDIV If Douglas method requested, divide N by'/
     +              T20,'   NDIV.  NDIV=5 is recommended.'/)

C       U1-U12 - User may use these to input data required by user
C                - routines - real - in common block. Their is a
C                 variable:  U0 which is not used anywhere
      U1=1
      WRITE(NCFG,160) U1
160     FORMAT(F10.3,T20,'NDAY=U1, # days in data set.'
     +                  T20,'Set U1=1 for ocean model data set'/)

      U2=64.99
      WRITE(NCFG,162) U2
162     FORMAT(F10.3,T20,'SLAT0=U2, latitude of starting field'/)

      U3=40.69
      WRITE(NCFG,164) U3
164     FORMAT(F10.3,T20,'SLNG0=U3, longitude of starting field'/)

      U4=0
      WRITE(NCFG,166) U4
166     FORMAT(F10.3,T20,'DIR=U4, direction of propagation of center
ray'/)

      U5=1.5
      WRITE(NCFG,168) U5
168     FORMAT(F10.3,T20,'BOTRHO=U5, density in bottom'/)

      U6=0.0
      WRITE(NCFG,170) U6
170     FORMAT(F10.3,T20,'BOTRHOG=U6, density gradient in bottom'/)

      U7=0.5
      WRITE(NCFG,172) U7
172     FORMAT(F10.3,T20,'BOTBETA=U7, attenuation in bottom'/)

      U8=0.0
      WRITE(NCFG,174) U8
174     FORMAT(F10.3,T20,'BOTBETAG=U8, attenuation gradient in bottom'/)
```

```
      U9=0.965
      WRITE(NCFG,176) U9
176     FORMAT(F10.3,T20,'CWCB=U9, SS ratio at bottom interface,
CW/CB'/)

      U10=1.7
      WRITE(NCFG,178) U10
178     FORMAT(F10.3,T20,'CGRAD=U10, sound speed gradient in bottom'/)

      U11=200.0
      WRITE(NCFG,180) U11
180     FORMAT(F10.3,T20,'SEDZ=U11, sediment thickness'/)

      U12=0.0
      WRITE(NCFG,182) U12
182     FORMAT(F10.3,T20,'U12 = spare, currently unused')

      CLOSE(UNIT=NCFG,STATUS='KEEP')
      RETURN
      END

      SUBROUTINE READ_CONFIG(TITLE,NDIM,FRQ,ZS,C0,ISF,RA,ZA,N,IHNK,
     +                       ITYPES,ITYPEB,ITYPPW,ITYPSW,FLDW,NSEC,
     +                       RMAX,DR,WDR,WZ1,WZ2,WDZ,WDTH,PDR,PDZ,
     +                       PDTH,ISFLD,ISVP,IBOT,DOUGRA,NDIV,
     +                       U1,U2,U3,U4,U5,U6,U7,U8,U9,U10,U11,U12)
C***********************************************************************
C   This subroutine reads the 'harvard.cfg' file if one exists.
C
C   It then reformats the information and generates file harvard.in
C   to make the FOR3D program a happy camper.
C***********************************************************************
      INTEGER NCFG
      PARAMETER (NCFG=19)
      CHARACTER TITLE*80
C***********************************************************************
      OPEN(NCFG,FILE='harvard.cfg',STATUS='OLD',FORM='FORMATTED')
      PRINT *
      PRINT *,'Now reading Configuration file'
      PRINT *

      READ(NCFG,90) TITLE
90      FORMAT(A80,16(/))

      READ(NCFG,*) NDIM
      DO I=1,4
        READ(NCFG,*)
      ENDDO

      READ(NCFG,*) FRQ
      READ(NCFG,*)

      READ(NCFG,*) ZS
      READ(NCFG,*)

      READ(NCFG,*) C0
```

```
DO I=1,2
  READ(NCFG,*)
ENDDO

READ(NCFG,*) ISF
DO I=1,6
  READ(NCFG,*)
ENDDO

READ(NCFG,*) RA
DO I=1,5
  READ(NCFG,*)
ENDDO

READ(NCFG,*) ZA
DO I=1,13
  READ(NCFG,*)
ENDDO

READ(NCFG,*) N
DO I=1,4
  READ(NCFG,*)
ENDDO

READ(NCFG,*) IHNK
DO I=1,6
  READ(NCFG,*)
ENDDO

READ(NCFG,*) ITYPES
DO I=1,4
  READ(NCFG,*)
ENDDO

READ(NCFG,*) ITYPEB
DO I=1,6
  READ(NCFG,*)
ENDDO

READ(NCFG,*) ITYPPW
DO I=1,4
  READ(NCFG,*)
ENDDO

READ(NCFG,*) ITYPSW
DO I=1,4
  READ(NCFG,*)
ENDDO

READ(NCFG,*) FLDW
READ(NCFG,*)

READ(NCFG,*) NSEC
DO I=1,2
  READ(NCFG,*)
ENDDO
```

```
READ(NCFG,*)RMAX
READ(NCFG,*)

READ(NCFG,*) DR
DO I=1,4
  READ(NCFG,*)
ENDDO

READ(NCFG,*) WDR
DO I=1,2
  READ(NCFG,*)
ENDDO

READ(NCFG,*) WZ1
READ(NCFG,*)

READ(NCFG,*) WZ2
DO I=1,2
  READ(NCFG,*)
ENDDO

READ(NCFG,*)WDZ
DO I=1,4
  READ(NCFG,*)
ENDDO

READ(NCFG,*) WDTH
DO I=1,2
  READ(NCFG,*)
ENDDO

READ(NCFG,*) PDR
DO I=1,2
  READ(NCFG,*)
ENDDO

READ(NCFG,*) PDZ
DO I=1,2
  READ(NCFG,*)
ENDDO

READ(NCFG,*) PDTH
DO I=1,2
  READ(NCFG,*)
ENDDO

READ(NCFG,*) ISFLD
DO I=1,3
  READ(NCFG,*)
ENDDO

READ(NCFG,*) ISVP
DO I=1,3
  READ(NCFG,*)
ENDDO

READ(NCFG,*) IBOT
```

```
      DO I=1,3
        READ(NCFG,*)
      ENDDO

      READ(NCFG,*) DOUGRA
      DO I=1,3
        READ(NCFG,*)
      ENDDO

      READ(NCFG,*) NDIV
      DO I=1,2
        READ(NCFG,*)
      ENDDO

C      U1-U12 - User may use these to input data required by user
C               - routines - real - in common block.

      READ(NCFG,*) U1
      READ(NCFG,*)
      READ(NCFG,*) U2
      READ(NCFG,*)
      READ(NCFG,*) U3
      READ(NCFG,*)
      READ(NCFG,*) U4
      READ(NCFG,*)
      READ(NCFG,*) U5
      READ(NCFG,*)
      READ(NCFG,*) U6
      READ(NCFG,*)
      READ(NCFG,*) U7
      READ(NCFG,*)
      READ(NCFG,*) U8
      READ(NCFG,*)
      READ(NCFG,*) U9
      READ(NCFG,*)
      READ(NCFG,*) U10
      READ(NCFG,*)
      READ(NCFG,*) U11

      CLOSE(UNIT=NCFG,STATUS='KEEP')
      RETURN
      END

      SUBROUTINE WRITE_HARVARD_IN(TITLE,NDIM,FRQ,ZS,C0,ISF,RA,ZA,N,IHNK,
     +                            ITYPES,ITYPEB,ITYPPW,ITYPSW,FLDW,NSEC,
     +                            RMAX,DR,WDR,WZ1,WZ2,WDZ,WDTH,PDR,PDZ,
     +                            PDTH,ISFLD,ISVP,IBOT,DOUGRA,NDIV,
     +                            U1,U2,U3,U4,U5,U6,U7,U8,U9,U10,U11,U12)
C***********************************************************************
C  Extract any data from FOR3D.IN necessary to build the output file
C  and query user for source location
C***********************************************************************
      PARAMETER(NFOR3D=17)
      CHARACTER TITLE*80
C-----------------------------------------------------------------------
      OPEN(UNIT=NFOR3D,FILE='harvard.in',FORM='FORMATTED',
     +     STATUS='UNKNOWN')
```

```
      WRITE(NFOR3D,15)TITLE
15    FORMAT(A80)

      WRITE(NFOR3D,20) NDIM
20      FORMAT(I1)

      WRITE(NFOR3D,30) FRQ,ZS,C0,ISF,RA,ZA,N,IHNK,ITYPES,ITYPEB,
     +             ITYPPW,ITYPSW,FLDW,NSEC

30      FORMAT(3(1X,F7.1),I7,2F7.1,I7,5(1X,I7),F7.2,I7)

      WRITE(NFOR3D,40) RMAX,DR,WDR,WZ1,WZ2,WDZ,WDTH,PDR,PDZ,PDTH,
     +             ISFLD,ISVP,IBOT
40      FORMAT(10(1X,F7.1),3(1X,I7))

      WRITE(NFOR3D,50) DOUGRA,NDIV
50      FORMAT(F7.1,I7/)

      WRITE(NFOR3D,60) U1,U2,U3,U4,U5,U6,U7,U8,U9,U10,U11,U12
60      FORMAT(12(1X,F7.3)/)

      WRITE(NFOR3D,70)
70      FORMAT('1'//'0'/'1')

      CLOSE(UNIT=NFOR3D)

      RETURN
      END

      SUBROUTINE YESNO(OKNO)
C***********************************************************************
C   This little operation occurs many times after prompt for Y/N
C***********************************************************************
      CHARACTER ANSWER*1
      LOGICAL OKNO

      ACCEPT 35, ANSWER
35      FORMAT (A)
      CALL CLEN(ANSWER,1,NCHAR)
      IF(NCHAR.EQ.0) THEN
        OKNO=.FALSE.
      ELSE
        OKNO= .NOT.(ANSWER.EQ.'Y' .OR. ANSWER.EQ.'y')
      ENDIF
      RETURN
      END

      SUBROUTINE CLEN(STRING,NMAX,NCHAR)
C***********************************************************************
C   Returns length of character string
C***********************************************************************
      CHARACTER STRING(NMAX)*1
C-----------------------------------------------------------------------
      NCHAR=NMAX
      DO WHILE (STRING(NCHAR) .EQ. ' ' .AND. NCHAR .GT. 0)
        NCHAR=NCHAR-1
```

```
          IF(NCHAR .EQ. 0) GOTO 99
        ENDDO

99      RETURN
        END
```

Appendix E.2 Plotting FOR3D Model Results

Program files 'make_plot_files.f'

```
C  make_plot_files.f     A. E. Leybourne    last revised 4/14/00
C**********************************************************************
C  This prograam reads the files created by for3d and extracts data to
C  be plotted by matlab scripts.  It is a front end designed to make the
C  the matlab scripts easier to write.
C**********************************************************************
      PARAMETER(MXNSEC=2751,NSOLMX=MXNSEC+1,NZMX=194)
      REAL*4  TMP1(NZMX),TMP2(NZMX),UMAG(NZMX)
      INTEGER*4 N_OUT(NSOLMX)
      LOGICAL STARTFILE/.TRUE./
      LOGICAL WRITE_OUTPUT_FILE(MXNSEC)
      CHARACTER STR1*1,STR80*80,NUMBR(NSOLMX)*2,STR4(NSOLMX)*4
      CHARACTER FILE_EXT*20
      DATA WRITE_OUTPUT_FILE/MXNSEC*.FALSE./

C   Load STR4 array with character file numbers
      DO I=1,NSOLMX
        ENCODE (4,5,STR4(I)) I
        DO J=1,3
          IF(STR4(I)(J:J) .EQ. ' ') STR4(I)(J:J) = '0'
        ENDDO
      ENDDO
5     FORMAT(I4)
C----------------------------------------------------------------------
C     Level to save in output file
      ISAVLEV=10
      N_IN=1

      OPEN(N_IN, FILE='HARVARD.OUT',STATUS='OLD',FORM='FORMATTED')

C   Read in the header

      READ(UNIT=N_IN,FMT=100,END=999) STR80
100   FORMAT(A)
      READ(UNIT=N_IN,FMT=*,END=999)
     +   NDIM, FRQ, ZS, C0, ISF, RA, ZA
         IF(NDIM .NE. 2 .AND. NDIM .NE. 3) THEN
           STOP 'Only NDIM = 2 or 3 allowed'
         ENDIF
      PRINT 200,STR80,
     +       NDIM, FRQ, ZS, C0, ISF, RA, ZA
200   FORMAT(/A/I3,3F10.2,I3,2F8.0)
      READ(UNIT=N_IN,FMT=100,END=999) STR80
      READ(UNIT=N_IN,FMT=100,END=999) STR80
      READ(UNIT=N_IN,FMT=*,END=999)
     +      N, IHNK,ITYPES, ITYPEB, ITYPPW, ITYPSW, FLDW
      PRINT 210,STR80,
     +      N, IHNK,ITYPES, ITYPEB, ITYPPW, ITYPSW, FLDW
210   FORMAT(/A/I5,5I7,F9.3)
      READ(UNIT=N_IN,FMT=100,END=999) STR80
      READ(UNIT=N_IN,FMT=100,END=999) STR80
      READ(UNIT=N_IN,FMT=*,END=999)
```

```
     +       NSEC, NSOL, RMAX, DR, WDR, WZ1, WZ2
      PRINT 220,STR80,
     +       NSEC, NSOL, RMAX, DR, WDR, WZ1, WZ2
 220  FORMAT(/A/I4,I5,F10.1,F5.1,3F7.1/)
      READ(UNIT=N_IN,FMT=100,END=999) STR80
      READ(UNIT=N_IN,FMT=100,END=999) STR80
      READ(UNIT=N_IN,FMT=*,END=999)
     +       WDZ, DZ, DOUGRA, NDIV,NSOLW,NRAD
      PRINT 230,STR80,
     +       WDZ, DZ, DOUGRA, NDIV, NSOLW,NRAD
 230  FORMAT(/A/2F8.2,F11.3,3I5)

      DO I=1,2
        READ(UNIT=N_IN,FMT=100,ERR=999) STR80
        PRINT 100, STR80
      ENDDO

      READ(UNIT=N_IN,FMT=*,ERR=999)NDAY,SLAT0,SLNG0,DIR,BOTRHO
      PRINT 240, NDAY,SLAT0,SLNG0,DIR,BOTRHO
 240  FORMAT(5(3X,F12.7))

      DO I=1,2
        READ(UNIT=N_IN,FMT=100,ERR=999) STR80
        PRINT 100, STR80
      ENDDO

      READ(UNIT=N_IN,FMT=*,ERR=999) BOTROHG,BOTBRTA,BOTBETAG,CWCB,CGRAD
      PRINT 250, BOTROHG,BOTBETA,BOTBETAG,CWCB,CGRAD
 250  FORMAT(5(3X,F12.7))

      DO I=1,2
        READ(UNIT=N_IN,FMT=100,ERR=999) STR80
        PRINT 100, STR80
      ENDDO

      READ(UNIT=N_IN,FMT=*,ERR=999) SEDZ,SPARE
      PRINT 260, SEDZ,SPARE
 260  FORMAT(5(3X,F12.7))

      DO I=1,3
        READ(UNIT=N_IN,FMT=100,ERR=999) STR80
        PRINT 100, STR80
      ENDDO
      PRINT *,   NSOL,NSOLMX

      IF(NSOL .GT. NSOLMX) STOP'Too many solutions required'

C   Determine which radials plot files to generate.
      PRINT *
      PRINT *,'When finished entering you selected radials to plot'
      PRINT *,'      enter 9999. Selection order may be random.'
      PRINT *
      PRINT *,'This file contains ',INT(NRAD),' Radials'
      DO WHILE (I.NE.9999)
        PRINT *,'Enter radial to plot ==> '
        ACCEPT *,I
        IF(I.LE.NRAD .AND. I.GE.1) WRITE_OUTPUT_FILE(I)=.TRUE.
```

```
      ENDDO

C   Get the name of the sound file that was used
      PRINT *
      PRINT *,'Enter the ?? part of the sound file name used, e.g.,'
      PRINT *,'the name   sound-3D.???? may have been sound-3D.grad10'
      PRINT *
      ACCEPT 35, STR80
35    FORMAT (A)
      CALL CLEN(STR80,80,NCHAR)
      FILE_EXT=STR80(1:NCHAR)
      PRINT *
      PRINT *,'You entered ==> ',FILE_EXT
      PRINT *


C   Open output radial files
      IF(NRAD.GT.NSOLMX)STOP 'Opening more than NSOLMX files not
supported'
      DO IRAD=1,NRAD
        IF(WRITE_OUTPUT_FILE(IRAD)) THEN
          N_OUT(IRAD)=IRAD+10
          IF(NDIM.EQ.2) THEN
            STR1='2'
          ELSE
            STR1='3'
          ENDIF
          OPEN(N_OUT(IRAD), FILE='rad_'//FILE_EXT(1:NCHAR)//'_'//
     +            STR1//'d'//STR4(IRAD)//
     +            '.plt',STATUS='UNKNOWN',FORM='FORMATTED')
        ENDIF
      ENDDO

      DO WHILE(0.EQ.0)
        DO I=1,2
C         This is the only READ statment that should reach EOF
          READ(UNIT=N_IN,FMT=100,ERR=999,END=998) STR80
          IF(STARTFILE)    PRINT 100, STR80
        ENDDO

        NZSAV=0
        DO IRAD=1,NRAD
          READ(UNIT=N_IN,FMT=*,ERR=999) ANG,NZ,RA,WZ1,WDZ,IWZ1,IWZ2,IWZ
C          PRINT 310, ANG,NZ,RA,WZ1,WDZ,IWZ1,IWZ2,IWZ
310       FORMAT(1X,F10.5,I6,F8.1,2F8.2,3I6)
          IF(NZ .GT. NZMX) STOP 'Too many depths required'
          IF(NZSAV.EQ.0) NZSAV=NZ
        IF(NZSAV.NE.0 .AND. NZ.NE.NZSAV) THEN
            STOP'Inconsistent NZ value in input file'
        ENDIF
          IF(STARTFILE) THEN
            IF(WRITE_OUTPUT_FILE(IRAD)) THEN
              WRITE(UNIT=N_OUT(IRAD),FMT=*)
     +            ANG,NZ,WZ1,WDZ,DZ,IWZ1,IWZ2,IWZ,SLAT0,SLNG0,DIR,
     +            BOTRHO,BOTROHG,BOTBETAG,CWCB,CGRAD,SEDZ,NDIM,FRQ,ZS
            ENDIF
          ENDIF
```

```
          READ(UNIT=N_IN,FMT=*,ERR=999)(TMP1(INZ),TMP2(INZ),INZ=1,NZ)
C          PRINT 320,(TMP1(INZ),TMP2(INZ),INZ=1,NZ)
320       FORMAT(2(E15.7,E15.7))
C         Save value to plot file work array
          DO INZ=1,NZ
             UMAG(INZ)= CABS(CMPLX(TMP1(INZ),TMP2(INZ)))
          ENDDO
C         Prevent writing trash values at the beginning of the file to
C         the file to be plotted. This avoids Matlab taking log of 0's.
          IF(RA .NE. 0) THEN
            IF(WRITE_OUTPUT_FILE(IRAD)) THEN
              WRITE(UNIT=N_OUT(IRAD),FMT=*) RA
              WRITE(UNIT=N_OUT(IRAD),FMT=*) (UMAG(INZ),INZ=1,NZ)
            ENDIF
          ENDIF
        ENDDO

        STARTFILE=.FALSE.
      ENDDO

998   CLOSE (UNIT=N_IN)
      DO IRAD=1,NRAD
       IF(WRITE_OUTPUT_FILE(IRAD)) THEN
C         Force an EOF flag value
        RA=-1
          WRITE (UNIT=N_OUT(IRAD),FMT=*) RA
          CLOSE (UNIT=N_OUT(IRAD))
        ENDIF
      ENDDO
      STOP'NORMAL EXIT'

999   STOP'FILE READ ERROR'
      END

      SUBROUTINE CLEN(STRING,NMAX,NCHAR)
C***********************************************************************
C   Returns length of character string
C***********************************************************************
      CHARACTER STRING(NMAX)*1
C-----------------------------------------------------------------------
      NCHAR=NMAX
      DO WHILE (STRING(NCHAR) .EQ. ' ' .AND. NCHAR .GT. 0)
        NCHAR=NCHAR-1
        IF(NCHAR .EQ. 0) GOTO 99
      ENDDO

99    RETURN
      END


      SUBROUTINE YESNO(OKNO)
C***********************************************************************
C   This little operation occurs many times after prompt for Y/N
C***********************************************************************
      CHARACTER ANSWER*1
      LOGICAL OKNO
```

```
      ACCEPT 35, ANSWER
35      FORMAT (A)
      CALL CLEN(ANSWER,1,NCHAR)
      IF(NCHAR.EQ.0) THEN
        OKNO=.FALSE.
      ELSE
        OKNO= .NOT.(ANSWER.EQ.'Y' .OR. ANSWER.EQ.'y')
      ENDIF
      RETURN
      END
```

Program file 'plot_range_at_depth.m'

```
% plot_range_at_depth.m    Last revised 6/18/99  A. E. Leybourne
%**********************************************************************
% This program is intended to plot the data contained in files
% rad_?d??.plt which are from a for3d_all_ocean.f run
%**********************************************************************

% Clear out all previous variables
  clear

% Inform this program of TL depth to plot
  disp(' ')
  disp('Enter TL profile depth figure is generated for')
  disp('at the nearest solution grid level')
  Plot_depth=input (' -->');

% Let user select file to plot
  filename=uigetfile('*.plt','Choose file to open.. ');

% Setup file channel number
  fid=eval(['fopen(filename,' '''r''' ')'])

% Get file 1st line of data
  UVARS    =fscanf(fid,'%e',[1,20])
  ANG      =UVARS(1,1);
  NZ       =UVARS(1,2);
  WZ1      =UVARS(1,3);
  WDZ      =UVARS(1,4);

  DZ       =UVARS(1,5);
  IWZ1     =UVARS(1,6);
  IWZ2     =UVARS(1,7);
  IWZ      =UVARS(1,8);

  SLAT0    =UVARS(1,9);
  SLNG0    =UVARS(1,10);
  DIR      =UVARS(1,11);
  BOTRHO   =UVARS(1,12);
  BOTROHG  =UVARS(1,13);
  BOTBETAG=UVARS(1,14);
  CWCB     =UVARS(1,15);
  CGRAD    =UVARS(1,16);
  SEDZ     =UVARS(1,17);
  NDIM     =UVARS(1,18);
  FRQ      =UVARS(1,19);
  ZS       =UVARS(1,20);

% Now read in the sound field
  irng=0;

  while (0==0)
    irng=irng+1;
    RA(irng,1) =fscanf(fid,'%e',[1,1]);
    if(RA(irng,1)==-1)
      RA=RA(1:irng-1,1);
```

```
      break
      string='eof reached'
    else
      U(irng,:)=fscanf(fid,'%e',[1,NZ]);
%     Convert magnitude of field to dB tranmission loss
      U(irng,:)=-20*log10(U(irng,:));
%     Adjust by 10 LOG r (Note this may have already been done by FOR3D)
      U(irng,:)=U(irng,:)+10*log10(RA(irng,1));
    end
  end

  string='Finished reading data file'

% Invert the field from as read in orientation prior to plotting
  U=U';

% Provide axis values
  Depth=IWZ1*DZ:WDZ:IWZ2*DZ;
  Range=RA/1000.;

  LPlot_index=floor((Plot_depth-IWZ1*DZ)/WDZ)+1
  HPlot_index=LPlot_index+1

  LPltD=Depth(LPlot_index)
  HPltD=LPltD+WDZ
  if (Plot_depth - LPltD) > (HPltD - Plot_depth)
    Plot_index=HPlot_index
  else
    Plot_index=LPlot_index
  end
  PltD=Depth(Plot_index)

  Slice98=U(Plot_index,:);

% Plot the data just read in
  h=figure(1);
  set(h,'PaperUnits','inches');
  set(h,'Units','Inches','PaperOrientation','landscape');
  set(h,'PaperPosition',[.75,.75,8.75,5.75]);
  set(h,'Position',[.75,.75,8.75,5.75]);


  imagesc(Range,Depth,U);
  set(gca,'fontsize',14)
  colormap(hot);
  caxis([40,100]);
  hc=colorbar
  set(hc,'fontsize',14)
  color_label=...
  text(1.23,0.25,'TRANSMISSION LOSS, dB','unit','norm','rotation',90,...
      'fontsize',14);

% x-axis label
  xlabel('RANGE FROM SOURCE, KM','fontsize',14);

% y-axis labels
  ylabel('DEPTH FROM SURFACE, M','fontsize',14);
```

```
% Title
  title(['FOR3D MODEL (' num2str(NDIM),'D CASE)  FREQ =',...
         num2str(FRQ),' HZ, RADIAL = ' num2str(ANG),' DEG'],...
         'fontsize',14);

%  text(0.04,0.95,['SOURCE: DEPTH = ' num2str(ZS) ' M, LAT =',...
%                  num2str(SLAT0),' DEG,  LONG =' num2str(SLNG0),...
%                  ' DEG'],'units','normalized','fontsize',14)

% save the plot
  outname=[filename(1:max(size(filename))-4) '.tif']
  eval(['print -zbuffer -dtiff ', outname])

%  eval(['!lpr -PDiego -s' outname])

% Plot the data just read in
  h=figure(2);

  set(h,'PaperUnits','inches');
  set(h,'Units','Inches','PaperOrientation','landscape');
  set(h,'PaperPosition',[.75,.75,8.75,5.75]);
  set(h,'Position',[.75,.75,8.75,5.75]);


  plot(Range,Slice98)
  h=gca;
  set(h,'Ydir','reverse')
  set(h,'Ylim',[40,100],'fontsize',14)
  % Title
  title(['FOR3D MODEL (' num2str(NDIM) 'D) TL @ DEPTH =' ...
          num2str(PltD) ' M, FREQ = ' ...
          num2str(FRQ) ' HZ, RADIAL = ' num2str(ANG),...
          ' DEG'],'fontsize',14);

  text(0.07,0.95,['SOURCE:   DEPTH = ' num2str(ZS) ' M, LAT = ',...
                  num2str(SLAT0),' DEG,  LONG = ' num2str(SLNG0),...
                  ' DEG'],'units','normalized','fontsize',13)

% X-axis label
  xlabel('RANGE FROM SOURCE, KM','fontsize',14);

% Y-axis label
  ylabel(['TRANSMISSION LOSS, dB'],'fontsize',14);

% save the plot
  outname=[filename(1:max(size(filename))-4) 'tl.tif']
  eval(['print ',outname,' -dtiff'])
```

ovly_plots_range_at_depth.m

```
% ovly_plots_range_at_depth.m    Last revised 5/16/00  A. E. Leybourne
%*********************************************************************
% This program is intended to plot the data contained in files
% rad_?d??.plt which are from a for3d_all_ocean.f run
%
% In the first pass, it reads in and plots the 2D mode file
% In the secnd pass, it reads in and over-plots the 3D mode file
%*********************************************************************

% Clear out all previous variables
  clear

% Inform this program of TL depth to plot
  disp(' ')
  disp('Enter TL profile depth figure is generated for')
  disp('at the nearest solution grid level')
  Plot_depth=input (' -->');

  for pass=1:1:2
  disp(' ')
  if pass==1
    disp('In this pass, select a 2D file to plot ')
  else
    disp('In this pass, select a 3D file to over-plot ')
    hold on
   end
  disp(' ')

% Let user select file to plot
  filename=uigetfile('*.plt','Choose file to open.. ');

% Setup file channel number
  fid=eval(['fopen(filename,' '''r''' ')'])

% Get file 1st line of data
  UVARS    =fscanf(fid,'%e',[1,20])
  ANG      =UVARS(1,1);
  NZ       =UVARS(1,2);
  WZ1      =UVARS(1,3);
  WDZ      =UVARS(1,4);

  DZ       =UVARS(1,5);
  IWZ1     =UVARS(1,6);
  IWZ2     =UVARS(1,7);
  IWZ      =UVARS(1,8);

  SLAT0    =UVARS(1,9);
  SLNG0    =UVARS(1,10);
  DIR      =UVARS(1,11);
  BOTRHO   =UVARS(1,12);
  BOTROHG  =UVARS(1,13);
  BOTBETAG=UVARS(1,14);
  CWCB     =UVARS(1,15);
  CGRAD    =UVARS(1,16);
```

```
  SEDZ     =UVARS(1,17);
  NDIM     =UVARS(1,18);
  FRQ      =UVARS(1,19);
  ZS       =UVARS(1,20);

% Now read in the sound field
  irng=0;
  clear RA
  clear U
  while (0==0)
    irng=irng+1;
    RA(irng,1) =fscanf(fid,'%e',[1,1]);
    if(RA(irng,1)==-1)
      RA=RA(1:irng-1,1);
      break
      string='eof reached'
    else
      U(irng,:)=fscanf(fid,'%e',[1,NZ]);
%     Convert magnitude of field to dB tranmission loss
      U(irng,:)=-20*log10(U(irng,:));
%     Adjust by 10 LOG r (Note this may have already been done by FOR3D)
      U(irng,:)=U(irng,:)+10*log10(RA(irng,1));
    end
  end

  string='Finished reading data file'

% Invert the field from as read in orientation prior to plotting
  U=U';

% Provide axis values
  Depth=IWZ1*DZ:WDZ:IWZ2*DZ;
  Range=RA/1000.;

  LPlot_index=floor((Plot_depth-IWZ1*DZ)/WDZ)+1
  HPlot_index=LPlot_index+1

  LPltD=Depth(LPlot_index)
  HPltD=LPltD+WDZ
  if (Plot_depth - LPltD) > (HPltD - Plot_depth)
    Plot_index=HPlot_index
  else
    Plot_index=LPlot_index
  end
  PltD=Depth(Plot_index)

  Slice98=U(Plot_index,:);


% Plot the data just read in
  if pass==1
    h=figure(1);
    set(h,'PaperUnits','inches');
    set(h,'Units','Inches','PaperOrientation','landscape');
    set(h,'PaperPosition',[.75,.75,8.75,5.75]);
    set(h,'Position',[.75,.75,8.75,5.75]);
  end
```

```
  if NDIM==2
    plot(Range,Slice98,':','linewidth',3)
  end

  if NDIM==3
    plot(Range,Slice98,'-')
  end

  if pass==1
    h=gca;
    set(h,'Ydir','reverse')
    set(h,'Ylim',[40,100],'fontsize',14)

% Title
    title(['FOR3D MODEL, TL @ DEPTH =' ...
          num2str(PltD) ' M, FREQ = ' ...
          num2str(FRQ) ' HZ, RADIAL = ' ...
          num2str(ANG) ' DEG'],'fontsize',14);

    text(0.07,0.95,['SOURCE:   DEPTH = ' num2str(ZS) ...
                     ' M, LAT = ' num2str(SLAT0) ...
         ' DEG,  LONG = ' num2str(SLNG0) ' DEG'],...
         'units','normalized','fontsize',13)

%   X-axis label
    xlabel('RANGE FROM SOURCE, KM','fontsize',14);

%   Y-axis label
    ylabel(['TRANSMISSION LOSS, dB'],'fontsize',14);

%   Compute and plot Legend Line
    text(0.02,0.085,'2D','units','normalized','fontsize',13)
    text(0.02,0.035,'3D','units','normalized','fontsize',13)
    hold on
      [atmp btmp]=size(Range);
      Xleg=[1 3]*(Range(atmp)/15);
      clear atmp btmp
      Yleg=[98 98];
      plot(Xleg,Yleg,'-')
      Yleg=[95 95];
      plot(Xleg,Yleg,':','linewidth',3)
    hold off
  end
  end  % end of pass loop

% save the plot
  newfilename=[filename(1:4) 'ovly_' filename(7:10)]
  outname=[newfilename '_dp' num2str(PltD) '.tif']
  eval(['print ',outname,' -dtiff'])
```

ovly_plots_depth_at_range.m

```
% ovly_plots_depth_at_range.m    Last revised 5/16/00  A. E. Leybourne
%*************************************************************************
% This program is intended to plot the data contained in files
% rad_?d??.plt which are from a for3d_all_ocean.f run
%
% In the first pass, it reads in and plots the 2D mode file
% In the secnd pass, it reads in and over-plots the 3D mode file
%*************************************************************************

% Clear out all previous variables
  clear

% Inform this program of TL Range to cross plot
  disp(' ')
  disp('Enter Range to cross plot that figure is generated for')
  disp('at the nearest solution grid range')
  Plot_range=input (' -->');

  for pass=1:1:2
  disp(' ')
  if pass==1
    disp('In this pass, select a 2D file to plot ')
  else
    disp('In this pass, select a 3D file to over-plot ')
    hold on
   end
  disp(' ')

% Let user select file to plot
  filename=uigetfile('*.plt','Choose file to open.. ');

% Setup file channel number
  fid=eval(['fopen(filename,' '''r''' ')'])

% Get file 1st line of data
  UVARS   =fscanf(fid,'%e',[1,20]);
  ANG     =UVARS(1,1);
  NZ      =UVARS(1,2);
  WZ1     =UVARS(1,3);
  WDZ     =UVARS(1,4);

  DZ      =UVARS(1,5);
  IWZ1    =UVARS(1,6);
  IWZ2    =UVARS(1,7);
  IWZ     =UVARS(1,8);

  SLAT0   =UVARS(1,9);
  SLNG0   =UVARS(1,10);
  DIR     =UVARS(1,11);
  BOTRHO  =UVARS(1,12);
  BOTROHG =UVARS(1,13);
  BOTBETAG=UVARS(1,14);
  CWCB    =UVARS(1,15);
  CGRAD   =UVARS(1,16);
```

```
  SEDZ     =UVARS(1,17);
  NDIM     =UVARS(1,18);
  FRQ      =UVARS(1,19);
  ZS       =UVARS(1,20);

% Now read in the sound field
  irng=0;
  clear RA
  clear U
  while (0==0)
    irng=irng+1;
    RA(irng,1) =fscanf(fid,'%e',[1,1]);
    if(RA(irng,1)==-1)
      RA=RA(1:irng-1,1);
      break
      string='eof reached'
    else
      U(irng,:)=fscanf(fid,'%e',[1,NZ]);
%     Convert magnitude of field to dB tranmission loss
      U(irng,:)=-20*log10(U(irng,:));
%     Adjust by 10 LOG r (Note this may have already been done by FOR3D)
      U(irng,:)=U(irng,:)+10*log10(RA(irng,1));
    end
  end

  string='Finished reading data file'

% Invert the field from as read in orientation prior to plotting
  U=U';

% Provide axis values
  Depth=IWZ1*DZ:WDZ:IWZ2*DZ;
  Range=RA/1000.;

% Adjust the RA requested to one that is in the file
  [temp1 temp2]=size(RA);
  RAinc=RA(temp1,1)/temp1;
%%%%%%%%%%%%%%%%%%%%%%%%%%
  for irange=1:1:temp1
    if RA(irange,1) <= Plot_range
       Plot_index = irange;
       PltRA=RA(Plot_index);
    end
  end
  Slice98=U(:,Plot_index);
  Slice98=Slice98';            % Flip index


% Plot the data just read in
  if pass==1
    h=figure(1);
    set(h,'PaperUnits','inches');
    set(h,'Units','Inches','PaperOrientation','landscape');
    set(h,'PaperPosition',[.75,.75,8.75,5.75]);
    set(h,'Position',[.75,.75,8.75,5.75]);
  end
```

```
if NDIM==2
  plot(Depth,Slice98,':','linewidth',3)
end

if NDIM==3
  plot(Depth,Slice98,'-')
end

if pass==1
  h=gca;
  set(h,'Ydir','reverse')
  set(h,'Ylim',[40,100],'fontsize',14)

% Title
  title(['FOR3D MODEL, TL @ RANGE =' ...
        num2str(PltRA) ' M, FREQ = ' ...
        num2str(FRQ) ' HZ, RADIAL = ' num2str(ANG) ,...
        ' DEG'],'fontsize',14);

  text(0.07,0.95,['SOURCE:   DEPTH = ' num2str(ZS) ,...
                ' M, LAT = ' num2str(SLAT0) ...
                ' DEG,  LONG = ' num2str(SLNG0) ' DEG'],...
                'units','normalized','fontsize',13)

%   X-axis label
    xlabel('DEPTH FROM SURFACE, M','fontsize',14);

%   Y-axis label
    ylabel(['TRANSMISSION LOSS, dB'],'fontsize',14);

%   Compute and plot Legend Line
    text(0.02,0.085,'2D','units','normalized','fontsize',13)
    text(0.02,0.035,'3D','units','normalized','fontsize',13)
    hold on
      [atmp btmp]=size(Depth);
      Xleg=[1 3]*(Depth(btmp)/15);
      clear atmp btmp
      Yleg=[98 98];
      plot(Xleg,Yleg,'-')
      Yleg=[95 95];
      plot(Xleg,Yleg,':','linewidth',3)
    hold off
  end
  end  % end of pass loop

% save the plot
  newfilename=[filename(1:4) 'ovly_' filename(7:10)]
  outname=[newfilename '_rg' num2str(PltRA) '.tif']
  eval(['print ',outname,' -dtiff'])
```

Appendix E.3 Beamforming of FOR3D Model Results

make_beamer_file.f

```
C  make_beamer_file.f     A. E. Leybourne    last revised 8/11/99
C*********************************************************************
C  This program reads HARVARD.OUT created by for3d and extracts data to
C  be beamformed by matlab scripts.  It is a front end designed to make
C  the matlab scripts easier to write.
C*********************************************************************
      PARAMETER(MXNSEC=2751,NSOLMX=MXNSEC+1,NZMX=194)
      PARAMETER(N_IN=11,N_OUT=12)
C        N_IN  Input file channel number
C        N_OUT Output file channel number
      REAL*4  TMP1(NZMX),TMP2(NZMX)
      COMPLEX PHONE(NSOLMX)
      LOGICAL STARTFILE/.TRUE./
      CHARACTER STR1*1,STR80*80,NUMBR(NSOLMX)*2
      CHARACTER DATAPATH*80
C---------------------------------------------------------------------
      DATAPATH=
     +'~/for3d/3d_ocean/ocean_data/current_trials/h150grad10-
1ft/3d/HARVARD.OUT'
      PRINT *,'File HARVARD.OUT must be in path below or change path'

      CALL CLEN(DATAPATH,80,NCHAR)
      IF(NCHAR .GE. 1) THEN
         PRINT *, DATAPATH(1:NCHAR)
      ELSE
         STOP 'DATA FILE NAME ERROR'
      ENDIF

      OPEN(N_IN, FILE=DATAPATH(1:NCHAR),STATUS='OLD',FORM='FORMATTED')
      OPEN(N_OUT, FILE='beam.in',STATUS='UNKNOWN',FORM='FORMATTED')

C   Read in the header and copy to the output file
      DO I=1,20
        READ(UNIT=N_IN,FMT=100,END=999) STR80
        CALL CLEN(STR80,80,NCHAR)
        IF(NCHAR .GE. 1) THEN
          WRITE(UNIT=N_OUT,FMT=*) STR80(1:NCHAR)
        ELSE
          WRITE(UNIT=N_OUT,FMT=*) ' '
        ENDIF
      ENDDO
      REWIND N_IN

C   Now Reread in the header details
      READ(UNIT=N_IN,FMT=100,END=999) STR80
100   FORMAT(A)
      READ(UNIT=N_IN,FMT=*,END=999)
     +   NDIM, FRQ, ZS, C0, ISF, RA, ZA
         IF(NDIM .NE. 2 .AND. NDIM .NE. 3) THEN
           STOP 'Only NDIM = 2 or 3 allowed'
         ENDIF
      PRINT 200,STR80,
```

```
     +        NDIM, FRQ, ZS, C0, ISF, RA, ZA
200   FORMAT(/A/I3,3F10.2,I3,2F8.0)
      READ(UNIT=N_IN,FMT=100,END=999) STR80
      READ(UNIT=N_IN,FMT=100,END=999) STR80
      READ(UNIT=N_IN,FMT=*,END=999)
     +        N, IHNK,ITYPES, ITYPEB, ITYPPW, ITYPSW, FLDW
      PRINT 210,STR80,
     +        N, IHNK,ITYPES, ITYPEB, ITYPPW, ITYPSW, FLDW
210   FORMAT(/A/I5,5I7,F9.3)
      READ(UNIT=N_IN,FMT=100,END=999) STR80
      READ(UNIT=N_IN,FMT=100,END=999) STR80
      READ(UNIT=N_IN,FMT=*,END=999)
     +        NSEC, NSOL, RMAX, DR, WDR, WZ1, WZ2
      PRINT 220,STR80,
     +        NSEC, NSOL, RMAX, DR, WDR, WZ1, WZ2
220   FORMAT(/A/I4,I5,F10.1,F5.1,3F7.1/)
      READ(UNIT=N_IN,FMT=100,END=999) STR80
      READ(UNIT=N_IN,FMT=100,END=999) STR80
      READ(UNIT=N_IN,FMT=*,END=999)
     +        WDZ, DZ, DOUGRA, NDIV,NSOLW,NRAD
      PRINT 230,STR80,
     +        WDZ, DZ, DOUGRA, NDIV, NSOLW,NRAD
230   FORMAT(/A/2F8.2,F11.3,3I5)

      DO I=1,2
        READ(UNIT=N_IN,FMT=100,ERR=999) STR80
        PRINT 100, STR80
      ENDDO

      READ(UNIT=N_IN,FMT=*,ERR=999)NDAY,SLAT0,SLNG0,DIR,BOTRHO
      PRINT 240, NDAY,SLAT0,SLNG0,DIR,BOTRHO
240   FORMAT(5(3X,F12.7))

      DO I=1,2
        READ(UNIT=N_IN,FMT=100,ERR=999) STR80
        PRINT 100, STR80
      ENDDO

      READ(UNIT=N_IN,FMT=*,ERR=999) BOTROHG,BOTBRTA,BOTBETAG,CWCB,CGRAD
      PRINT 250, BOTROHG,BOTBETA,BOTBETAG,CWCB,CGRAD
250   FORMAT(5(3X,F12.7))

      DO I=1,2
        READ(UNIT=N_IN,FMT=100,ERR=999) STR80
        PRINT 100, STR80
      ENDDO

      READ(UNIT=N_IN,FMT=*,ERR=999) SEDZ,SPARE
      PRINT 260, SEDZ,SPARE
260   FORMAT(5(3X,F12.7))

      DO I=1,3
        READ(UNIT=N_IN,FMT=100,ERR=999) STR80
        PRINT 100, STR80
      ENDDO
      PRINT *,    NSOL,NSOLMX
```

```
      IF(NSOL .GT. NSOLMX) STOP'Too many solutions required'
C%%%%%%%% make an interactive prompt for beamdepth
      ILEVTOSAV=100
      BEAMDEPTH=WZ1+WDZ*(ILEVTOSAV-1)

C  This routine is not very elegant, the entire file is read in just
C  to be able to save the data from the last range, RA in the file
      DO WHILE(0.EQ.0)
        DO I=1,2
C         This is the only READ statment that should legimately reach
EOF
          READ(UNIT=N_IN,FMT=100,ERR=999,END=990) STR80
          IF(STARTFILE)    PRINT 100, STR80
        ENDDO

        NZSAV=0
        DO IRAD=1,NRAD
          READ(UNIT=N_IN,FMT=*,ERR=999) ANG,NZ,RA,WZ1,WDZ,IWZ1,IWZ2,IWZ
C          PRINT 310, ANG,NZ,RA,WZ1,WDZ,IWZ1,IWZ2,IWZ
310       FORMAT(1X,F10.5,I6,F8.1,2F8.2,3I6)
          IF(NZ .GT. NZMX) STOP 'Too many depths required'
          IF(NZSAV.EQ.0) NZSAV=NZ
        IF(NZSAV.NE.0 .AND. NZ.NE.NZSAV) THEN
            STOP'Inconsistent NZ value in input file'
        ENDIF
          IF(STARTFILE .AND. IRAD.EQ.1) THEN
            WRITE(UNIT=N_OUT,FMT=*)
            WRITE(UNIT=N_OUT,FMT=*) '     ANG        NZ       WZ1     ',
     +             '    WDZ    IWZ1 IWZ2   IWZ'
            WRITE(UNIT=N_OUT,FMT=315) ANG,NZ,WZ1,WDZ,IWZ1,IWZ2,IWZ
315         FORMAT(F10.5,I7,2F10.5,3I5/)
            PRINT *,'ILEVTOSAV, BEAMDEPTH ',ILEVTOSAV,BEAMDEPTH
          ENDIF
          READ(UNIT=N_IN,FMT=*,ERR=999)(TMP1(INZ),TMP2(INZ),INZ=1,NZ)
C          PRINT 320,(TMP1(INZ),TMP2(INZ),INZ=1,NZ)
320       FORMAT(2(E15.7,E15.7))
C         Save value to beamer file work array
          PHONE(IRAD)=CMPLX(TMP1(ILEVTOSAV),TMP2(ILEVTOSAV))
        ENDDO
        STARTFILE=.FALSE.
      ENDDO
990   WRITE (UNIT=N_OUT,FMT=*) '    NRAD    RA_LAST    BEAMDEPTH'
      WRITE (UNIT=N_OUT,FMT=992) NRAD,RA,BEAMDEPTH
992   FORMAT(I7,2F10.2/)
      DO IRAD=1,NRAD

        WRITE (UNIT=N_OUT,FMT=*) REAL(PHONE(IRAD)),AIMAG(PHONE(IRAD))
994     FORMAT(2E14.7)
      ENDDO
      CLOSE (UNIT=N_IN)
      CLOSE (UNIT=N_OUT)

      STOP'NORMAL EXIT'

999   STOP'FILE READ ERROR'
      END
```

```
      SUBROUTINE CLEN(STRING,NMAX,NCHAR)
C***********************************************************************
C   Returns length of character string
C***********************************************************************
      CHARACTER STRING(NMAX)*1
C-----------------------------------------------------------------------
      NCHAR=NMAX
      DO WHILE (STRING(NCHAR) .EQ. ' ' .AND. NCHAR .GT. 0)
        NCHAR=NCHAR-1
        IF(NCHAR .EQ. 0) GOTO 99
      ENDDO

99    RETURN
      END


      SUBROUTINE YESNO(OKNO)
C***********************************************************************
C   This little operation occurs many times after prompt for Y/N
C***********************************************************************
      CHARACTER ANSWER*1
      LOGICAL OKNO

      ACCEPT 35, ANSWER
35      FORMAT (A)
      CALL CLEN(ANSWER,1,NCHAR)
      IF(NCHAR.EQ.0) THEN
        OKNO=.FALSE.
      ELSE
        OKNO= .NOT.(ANSWER.EQ.'Y' .OR. ANSWER.EQ.'y')
      ENDIF
      RETURN
      END
```

Program file 'beamer.m'

```
% File beamer.m      A. E. Leybourne III 6/26/2000
% ------------------------------------------------------------------
% The purpose of this program is to beamform the output of the FOR3D
% model at the maximum range in its output file HARVARD.OUT.  In order
% make this matlab script simpler to write, the FORTRAN program,
% make_beaamer_file.f, is used to read the above file and extract the
% information needed for the beamformer.
%
% Most of the header variables in the file are transcribed so that the
% the output plots can be properly annotated, however, the original
% sound field filename must be entered by the user.  This information is
% only used to annotate the beamformed plot.
% ------------------------------------------------------------------
%
%              Taken from program beam_demo.f
% ------------------------------------------------------------------
% First a signal at lamda/2 spacing, theta angle of arrival, f hz
% is generated for beam forming.
%
%   c      = 1500                    % Sound speed m/sec
%   f      = 100                     % Signal Frequency
%   fd     = 200                     % Array Design frequency, Hz
%   w      = 2*pi*f                  % frequency, rad/sec
%   lambda = c/fd                    % Hz design wave length, m
%   d      = lambda/2                % distance between phones, m
%   N      = 256                     % number of hydrophones
%   theta  = pi*(5/180)              % angle between wave front and array
%   ldel   = d*sin(theta)            % xtra wave travel incr. each phone
%   mphone = 1:1:N;                  % phone numbers
%   ld     = ldel*(mphone-1);        % xtra wave travel dist.each phone
%   td     = ld/c;                   % time delay to each phone, sec, 0 - N
%   m      =1:1:N;                   % beam numbers
%   cosangm=(fd/f)*(1-2*(m-1)/N);    % beam angle function
%
% Remove virtual beams
%   for ibeam=1:1:N
%     if(cosangm(ibeam) > 1)
%       cosangm(ibeam)=1;
%     end
%
%     if(cosangm(ibeam) < -1)
%       cosangm(ibeam)=-1;
%     end
%   end
%
%   beamangle=asin(cosangm);         % beam angles radians
%   beamangle=beamangle*180/pi;      % beam angle degrees
%
%   A      = 1                       % Cosine wave magnitude
%   SR     = A*cos(w*td);            % Real pressure at each phone
%   SI     = A*sin(w*td);            % Imag pressure at each phone
%   S      = SR +i*SI;               % Complex pressure at each phone
% ------------------------------------------------------------------
clear
```

```
% Get source sound filename
   soundfile=input(['Enter name of the soundfield file used for the',...
   '\nFOR3D run enclosed in single quotes. \n==> ']);

% Let user select file to plot
   infile=uigetfile('*.in','Choose file to open.. ');

% Setup file channel number
   fid=eval(['fopen(infile,' '''r''' ')']);

% Read in 1st 3 lines from file beamer.in
   STRVAR =fscanf(fid,'%s',[1,7]);
   UVARS  =fscanf(fid,'%e',[1,7]);
   NDIM   =UVARS(1,1);
   FRQ    =UVARS(1,2);
   ZS     =UVARS(1,3);
   C0     =UVARS(1,4);
   ISF    =UVARS(1,5);
   RA     =UVARS(1,6);
   ZA     =UVARS(1,7);

% Read in 2nd lines from file beamer.in
   STRVAR =fscanf(fid,'%s',[1,7]);
   UVARS  =fscanf(fid,'%e',[1,7]);
   N      =UVARS(1,1);
   IHNK   =UVARS(1,2);
   ITYPES =UVARS(1,3);
   ITYPEB =UVARS(1,4);
   ITYPEPW=UVARS(1,5);
   ITYPESW=UVARS(1,6);
   FLDW   =UVARS(1,7);

% Read in 3rd 3 lines from file beamer.in
   STRVAR =fscanf(fid,'%s',[1,7]);
   UVARS  =fscanf(fid,'%e',[1,7]);
   NSECS  =UVARS(1,1);
   NSOL   =UVARS(1,2);
   RMAX   =UVARS(1,3);
   DR     =UVARS(1,4);
   WDR    =UVARS(1,5);
   WZ1    =UVARS(1,6);
   WZ2    =UVARS(1,7);

% Read in 4th 3 lines from file beamer.in
   STRVAR =fscanf(fid,'%s',[1,6]);
   UVARS  =fscanf(fid,'%e',[1,6]);
   WDZ    =UVARS(1,1);
   DZ     =UVARS(1,2);
   DOUGRA =UVARS(1,3);
   NDIV   =UVARS(1,4);
   NSOLW  =UVARS(1,5);
   NRAD   =UVARS(1,6);

% Read in 5th 3 lines from file beamer.in
   STRVAR =fscanf(fid,'%s',[1,5]);
   UVARS  =fscanf(fid,'%e',[1,5]);
   NDAY   =UVARS(1,1);
```

```
  SLAT0   =UVARS(1,2);
  SLNG0   =UVARS(1,3);
  DIR     =UVARS(1,4);
  BOTHRO  =UVARS(1,5);

% Read in 6th 3 lines from file beamer.in
  STRVAR  =fscanf(fid,'%s',[1,5]);
  UVARS   =fscanf(fid,'%e',[1,5]);
  BOTROHG =UVARS(1,1);
  BOTBETA =UVARS(1,2);
  BOTBETAG=UVARS(1,3);
  CWCB    =UVARS(1,4);
  CGRAD   =UVARS(1,5);

% Read in 6th 3 lines from file beamer.in
  STRVAR =fscanf(fid,'%s',[1,2]);
  UVARS  =fscanf(fid,'%e',[1,2]);
  SEDZ   =UVARS(1,1);
  SPARE  =UVARS(1,2);

% Read in 7th 3 lines from file beamer.in
  STRVAR =fscanf(fid,'%s',[1,7]);
  UVARS  =fscanf(fid,'%e',[1,7]);
  ANG    =UVARS(1,1);
  NZ     =UVARS(1,2);
  WZ1    =UVARS(1,3);
  WDZ    =UVARS(1,4);
  IWZ1   =UVARS(1,5);
  IWZ2   =UVARS(1,6);
  IWZ    =UVARS(1,7);

% Read in 8th 3 lines from file beamer.in
  STRVAR =fscanf(fid,'%s',[1,3]);
  UVARS  =fscanf(fid,'%e',[1,3]);
  NRAD        =UVARS(1,1);
  RALAST      =UVARS(1,2);
  BEAMDEPTH   =UVARS(1,3);

% Read in the (simulated) hydrophone data
  for iphone=1:1:NRAD
    tmp1=fscanf(fid,'%e14',[1,1]);
    tmp2=fscanf(fid,'%e14',[1,1]);
    S(1,iphone)=tmp1+i*tmp2;
  end

% Get signal phase and plot it
  P=atan2(imag(S),real(S))*180/pi;
  Smag=abs(S);

  figure(1);
  set(gcf,'PaperUnits','inches');
  set(gcf,'Units','Inches','PaperOrientation','landscape');

  set(gcf,'PaperPosition',[.75,.8,8.75,3.75]);
  set(gcf,'Position',[.75,.75,8.75,3.75]);
  set(gcf,'PaperUnits','inches','paperpositionmode','manual');
  set(gca,'Units','Inches')
```

```
set(gca,'Position',[.75,.75,7.75,2.5]);
set(gca,'Units','Normalized')
X=-FLDW/2:FLDW/NSECS:FLDW/2;
plot(X,Smag)
title(['FOR3D (',num2str(NDIM),'D) ',...
        'MAGNITUDE AT RANGE, ',num2str(RMAX/1000),...
        ' KM AND DEPTH, ',num2str(ZS),' M'],'fontsize',14)
ylabel('RELATIVE MAGNITUDE','fontsize',14);
xlabel('ANGULAR OFFSET FROM CENTER RADIAL, DEGREES','fontsize',14);
text(0.37,0.95,[' NO. RADIALS, ',num2str(NRAD)],...
                 'units','normalized','fontsize',13);
set(gca,'Ylim',[0,0.1],'fontsize',14);

% save the plot
outname=['beamer-',num2str(NDIM),'d-mag.tif']
eval(['print -zbuffer -dtiff ', outname])

figure(2);
set(gcf,'PaperUnits','inches');
set(gcf,'Units','Inches','PaperOrientation','landscape');

set(gcf,'PaperPosition',[.75,.8,8.75,3.75]);
set(gcf,'Position',[.75,.75,8.75,3.75]);
set(gcf,'PaperUnits','inches','paperpositionmode','manual');
set(gca,'Units','Inches')
set(gca,'Position',[.75,.75,7.75,2.5]);
set(gca,'Units','Normalized')

plot(X,P)

title(['FOR3D (',num2str(NDIM),'D) ',...
        'PHASE AT RANGE, ',num2str(RMAX/1000),...
        ' KM AND DEPTH, ',num2str(ZS),' M'],'fontsize',14);

text(0.37,0.95,[' NO. RADIALS, ',num2str(NRAD)],...
                 'units','normalized','fontsize',13)

xlabel('ANGULAR OFFSET FROM CENTER RADIAL, DEGREES','fontsize',14);
ylabel('PHASE, DEGREES','fontsize',14)

set(gca,'Ylim',[-200,250],'fontsize',14);

% save the plot
outname=['beamer-',num2str(NDIM),'d-phase.tif']
eval(['print -zbuffer -dtiff ', outname])

%  S is a simulated phone array at Range RALAST from the sound source,
%  with field width FLDW degrees and number of radials NRAD, therefore:
delangle=abs(FLDW*(pi/180)/(NRAD-1)); % In radians
d=RALAST*asin(delangle);              % Meters, approximately
lamda =2*d;                           % Design wavelength
fd = C0/lamda;                        % Design frequency
f=FRQ;                                % Signal frequency

% Force N to be a power of 2
N=fix(NRAD/2)*2;
```

```
   S=S(1:N);
   han     =hanning(N)';         % Hanning window for shading (flipped)
   han     =han*(N/sum(han));    % Normalize to preserve magnitude
   S=      han.*S;               % Shadded spatial complex array

% This next trick causes a broadside beam to be shifted such that the
% output center of the beam is at N/2+1 in the array, for N=power of 2
% angular offsets are then shifted relative to this position.
% This algorithm from Marshall Bradley of PSI in Slidell, 8/1/94
   Ssign(1,1)=1;
   for j=2:1:N
     Ssign(1,j) = -1*Ssign(j-1);
   end
   Smodified  = S.*Ssign;           % Modified complex array values


% In the situation of a beamformer, S would be the result of an FFT
% performed on the time domain arrival of the signal at the array. An
FFT
% FFT causes the magnitude to modified and therefore NEEDS re-
nomalizing.
% Since these values were synthesized, this is not required here.

% Form the beams, using an inverse FFT transform
   B=ifft(Smodified);
% Normalized to maximum beam = 1.0
   Y=abs(B);
   maxY=max(Y);
   Y=Y/maxY;

% Develop the beam angle function
   m=1:1:N;                       % beam numbers
   cosangm=(fd/f)*(1-2*(m-1)/N);  % beam angle function

% Remove virtual beams
   for ibeam=1:1:N
     if(cosangm(ibeam) > 1)
       cosangm(ibeam)=1;
     end

     if(cosangm(ibeam) < -1)
       cosangm(ibeam)=-1;
     end
   end

% Use asin instead of acos to make broadside beam angle 0
   beamangle=asin(cosangm);        % beam angles radians
   beamangle=beamangle*180/pi;     % beam angle degrees

   figure(3);
   set(gcf,'PaperUnits','inches');
   set(gcf,'Units','Inches','PaperOrientation','landscape');
   set(gcf,'PaperPosition',[.75,.8,8.75,3.75]);
   set(gcf,'Position',[.75,.75,8.75,3.75]);
   set(gcf,'PaperUnits','inches','paperpositionmode','manual');
   set(gca,'Units','Inches')
   set(gca,'Position',[.75,.75,7.75,2.5]);
```

```
 set(gca,'Units','Normalized')

 plot(beamangle,Y)

% x-axis label
 xlabel('BEAM ANGLE, DEGREES (BROADSIDE AT 0 DEGREES)','fontsize',14);

% y-axis label
 ylabel('MAGNITUDE NORMALIZED TO 1.0','fontsize',14);

% Title

 title(['BEAMFORMED FOR3D (',num2str(NDIM),'D) MODEL RESULTS FOR ',...
      num2str(f),' HZ SOURCE'],'fontsize',14);

 xloc=0.45;
 yloc=0.90;

% Add text
 text(xloc,       yloc,['SOUND FILE:'],'Units','Normalized',...
                       'fontsize',13);
 text(xloc+0.22,yloc,[soundfile],'Units','Normalized',...
                       'fontsize',13);
 text(xloc,       yloc-0.10,['WEDGE WIDTH ='],'Units','Normalized',...
                       'fontsize',13);
 text(xloc+0.22,yloc-0.10,[num2str(FLDW),' DEGREES'],'Units',...
                       'Normalized','fontsize',13);
 text(xloc,       yloc-0.17,['SOURCE DEPTH ='],'Units',...
                       'Normalized','fontsize',13);
 text(xloc+0.22,yloc-0.17,[num2str(ZS),' M'],'Units',...
                       'Normalized','fontsize',13);
 text(xloc,       yloc-0.24,['BEAMER DEPTH ='],'Units',...
                       'Normalized','fontsize',13);
 text(xloc+0.22,yloc-0.24,[num2str(BEAMDEPTH),' M'],'Units',...
                       'Normalized','fontsize',13);
 text(xloc,       yloc-0.31,['RANGE ='],'Units','Normalized',...
                       'fontsize',13);
 text(xloc+0.22,yloc-0.31,[num2str(RALAST/1000),' KM'],'Units',...
                       'Normalized','fontsize',13);
 text(xloc,       yloc-0.38,['ARRAY DESIGN='],'Units',...
                       'Normalized','fontsize',13);
 text(xloc+0.22,yloc-0.38,[num2str(fix(10*fd)/10),' HZ'],...
                       'Units','Normalized','fontsize',13);
 text(xloc,       yloc-0.45,['NO. PHONES='],'Units',...
                       'Normalized','fontsize',13);
 text(xloc+0.22,yloc-0.45,[num2str(NSOL)],'Units',...
                       'Normalized','fontsize',13);

 h=gca;
 set(h,'Xlim',[-10 10],'fontsize',14);

 % save the plot
 outname=['beamer-',num2str(NDIM),'d-beams.tif']
 eval(['print -zbuffer -dtiff ', outname])
```